Second Annual

OS/2 World Conference & Exhibition

Proceedings — Volume 1

July 17-20, 1995
Hynes Convention Center
Boston, Mass.

Miller Freeman
A United News & Media company

this page intentionally left blank

Second Annual

OS/2 World Conference & Exhibition

Proceedings — Volume 1

ISBN #0-87930-396-4

©Copyright 1993 by Miller Freeman, Inc., and as designated. All rights reserved. No part of this book may be reproduced or copied in any manner whatsoever without the express written consent of Miller Freeman, Inc.

Miller Freeman, Inc. • 600 Harrison Street • San Francisco, CA 94107 • (415) 905-2200

this page intentionally left blank

Table of Contents

List of Authors 9

LAN 0109 LAN Server User Tips and Techniques 15
Gary Hunt

MGT 0111 To GUI or to OOUI: That is the Question 25
Theo Mandel

MNC 0102 OS/2 Warps the Internet * 47
David Moskowitz

MNC 0110 IBM Personal Systems Services and Support * 49
Desiree Piccini

PRO 0101 Designing High Performance OS/2 Applications 51
David Reich

PRO 0104 Enhancing the Enhanced Editor 65
Bill Schindler

SYS 0105 How to Create a User-Friendly OS/2 GUI/OOUI 71
Luca Miccoli

SYS 0106 Communications Manager/2: Overview and Directions 79
George Sweeney

SYS 0107 Introduction to DCE Administration * 93
Gerry Thome

LAN 0204 Fine-Tuning OS/2 LAN Server 95
Len Brissette

LAN 0206 OS/2 Distributed Computing Environment Overview 109
Sanford Rockowitz

*No paper available; contact speaker for further information

MGT 0202 The New Face of Teamwork 131
Richard Sullivan

MM 0208 The OS/2 Multimedia Experience 139
Maria Ingold

PRO 0203 SOM Programming on OS/2 157
Hari Madduri

PRO 0205 Intro to REXX part 1 163
Chip Davis

PRO 0207 C++ Application Frameworks Under the Microscope 177
Larry O'Brien

SYS 0201 OS/2 for the Power PC, the IBM Microkernel Architecture: The Big Picture 189
Miles Barel

LAN 0305 Migrating from Netware to LAN Server: Why and How * 199
Steve Baumgarner

MGT 0303 OS/2 Strategy and Directions 201
Miles Barel

MM 0301 OS/2 Multimedia for Developers 207
Maria Ingold

MNC 0302 Installing the NetWare Client for OS/2 235
Bob Panza

MNC 0310 How to Get the Most Out of Your OS/2 Warp 245
David Moskowitz

PRO 0304 Intro to REXX part 2 253
Chip Davis

*No paper available; contact speaker for further information

PRO 0306 Introduction to SOM for C Programmers 271
Roger Sessions

PRO 0307 Storing Objects in DB/2: The Ultimate Object Datastore 287
Guylaine Cantin

SYS 0308 Inside OS/2, A Technical View 313
Ron Cadima

CS 0404 DB2 Family: Version 2 of DB2/2 and DB2/6000 323
Dan Britton

MGT 0409 OS/2 Warp, Journey into the Unknown 343
Ron Cadima

MGT 0411 Electronic Marketing: Profits & Pitfalls 355
Rick Chapman

MNC 0401 Unleashing OS/2: Getting the Top Performance Right Away * 363
David Moskowitz & David Reich

MNC 0406 TCP/IP for OS/2 365
Bob Panza

PRO 0407 Introduction to the SOMobjects Toolkit 377
Ira Forman

PRO 0408 Object REXX 397
Rick McGuire

SYS 0402 Is the PC Out of Gas? What's New on the Desktop? 407
Mark Minasi

SYS 0403 DB/2 and COBOL: Tools for Corporate Developers 417
Robert Chapman

*No paper available; contact speaker for further information

SYS 0405 LAN Server Administration Tips and Techniques 429
Gary Hunt

CS 0507 Basic Client/Server Programming with CPI-C 441
Dave Lutz

LAN 0501 LAN Server 4.0 in a Warp Environment 463
James Norman

LAN 0508 LAN Distance Overview 477
Bob Spickelmier

PRO 0509 IBM COBOL goes Object-Oriented 489
Ernie Escuton

PRO 0511 SOMobjects: Management Utilities for Distributed SOM 503
Bart Jacob

SYS 0505 Understanding OS/2's Multiprotocol Interoperability 513
Bob Panza

SYS 0506 Database Front Ends: Challenges, Standards and Rules 525
Alan Levin

CS 0605 CICS OS/2: 32-bit Client/Server Computing 551
Jim McNair

MGT 0604 GUI-OOUI Update: IBM's OS/2 Warp and Microsoft's Windows 95 User Interfaces 569
Theo Mandel

MGT 0608 IBM Object Strategy and Plans 589
Anthony Brown

PRO 0602 An Overview of Object COBOL for OS/2 595
Frank Coyle

*No paper available; contact speaker for further information

PRO 0609 Exploring Genetic Algorithms and Evolutionary Programming 601
Larry O'Brien

PRO 0610 C Set ++ Overview 615
Alistair Rennie

SYS 0606 DB2/2 Performance and Tuning 629
Rick Weaver

SYS 0607 NetSP Secured Network Gateway 651
Peter Crotty

CS 0704 SockCap: Client/Server Programs in Object REXX 661
Rick McGuire

*No paper available; contact speaker for further information

this page intentionally left blank

List of Authors

Phil Arthur
IBM Corp. 40-C3-02
1 East Kirkwood Boulevard
Roanoke TX 76299

Miles Barel
IBM
see Cathy Miller

Joyce Batty
IBM
see Cathy Miller

Steve Baumgarner
IBM
see Cathy Miller

Noel Bergman
Development Technology Inc.
308 Springwood Road
Forest Acres SC 29206-2113

Evan Bergman
Development Technology Inc.
308 Springwood Road
Forest cres SC 29206-2113

Edwin Black
OS/2 Professional Magazine
172 Rollins Avenue
Rockville MD 20852

Len Brissette
IBM
see Cathy Miller

Dan Britton
IBM
see Cathy Miller

Anthony Brown
IBM
see Cathy Miller

Ron Cadima
IBM
see Cathy Miller

Guylaine Cantin
IBM
see Cathy Miller

Robert Chapman
Chapman & Associates
11 Marcblu
Suite 150
Aliso Viejo CA 92656

Rick Chapman
Aegis
35 Ridge Road
Stratford CT 06497-3243

John Conner
IBM
see Cathy Miller

Frank Coyle
Computer Science Department
Southern Methodist University
Dallas TX 75275

Peter Crotty
IBM
Cathy Miller

Chip Davis
7254 Pommel Dr
Sykesville MD 21784-5931

Pete Devenyi
IBM
see Cathy Miller

Craig Elliott
IBM
see Cathy Miller

Ernie Escuton
IBM
see Cathy Miller

Chris Fierros
IBM
see Cathy Miller

Ira Forman
IBM
see Cathy Miller

Ira Friedman
IBM
see Cathy Miller

David Gaertner
IBM
see Cathy Miller

Ken Greenlee
IBM
see Cathy Miller

Bob Gress
IBM
see Cathy Miller

Tom Hadley
IBM Corporation
1001 Winstead Drive
Cary NC 27709

Bryan Heft
IBM
see Cathy Miller

Gary Hunt
IBM
see Cathy Miller

Maria Ingold
IBM
see Cathy Miller

Bart Jacob
IBM
Intern'l Technical Support Organization
11400 Burnet Rd.
Austin TX 78758

Daniel Kardell
One Up Corporation
1603 LBJ Freeway
Suite 200
Dallas TX 75234

Laura Jeanne Knapp
IBM Corporation
P O Box 12195 473/500
3039 Cornwallis Drive
Research Triangle Park NC 27709

Feite Kraay
One Up
1603 LBJ Freeway
Suite 200
Dallas TX 75234

Ron Langer
IBM
see Cathy Miller

Kelvin Lawrence
IBM
see Cathy Miller

Alan Levin
Decisions Support Technology
125 CambridgePark Drive
Cambridge MA 02140

Mike Logan
IBM
see Cathy Miller

Dave Lutz
IBM
see Cathy Miller

Hari Madduri
IBM—MS1002
11400 Burnet Road
Austin TX 78758

Theo Mandel
Interface Design and Development
8001 Two Coves Drive
Austin TX 78730-3125

Barbara Martens
see Charles Sweet

Rick McGuire
IBM
see Cathy Miller

Jim McNair
IBM
see Cathy Miller

Luca Miccoli
IBM Semea S.P.A.
Rome Networking Software Laboratory
P.le G. Pastore 6
00144 Roma
Italy

Mark Minasi
Mark Minasi & Company
5229 Wilson Blvd
Arlington VA 22205

Cathy Miller
IBM/PSP Trade Show Marketing
11400 Burnet Road
Austin TX 78758

John Montgomery
IBM Microelectronics Division D260
1701 North Street
Endicott NY 13760

David Moskowitz
Productivity Solutions
164 Avendale Road
Norristown PA 19403

Dale Nilsson
IBM
see Cathy Miller

James Norman
Norcom Consulting Inc.
104 Cameron Mews
Alexandria VA 22314

Larry O'Brien
Miller Freeman, Inc.
600 Harrison Street
San Francisco, CA 94107

Bob Panza
The Systems Expert Inc.
30 Hesseky Meadow Rd
Box 1067
Woodbury CT 06798

Desiree Piccini
IBM
see Cathy Miller

Brian Proffit
Visionary Research
2706 SW 4th Street
Boynton Beach FL 33435

David Reich
10299 186th Ct South
Boca Raton FL 33498

Alistair Rennie
IBM
see Cathy Miller

Don Richards
IBM
see Cathy Miller

Sanford Rockowitz
Minaret Software
1373 South Mayfair Avenue
Daly City CA 94015

Bill Rooks
IBM
see Cathy Miller

Bill Schindler
TGC
28824 North 57th Street
Cave Creek AZ 85331

Esther Schindler
TGC
28824 North 57th Street
Cave Creek AZ 85331

Roger Sessions
IBM
see Cathy Miller

David Sharp
IBM
see Cathy Miller

Bob Spickelmier
IBM
see Cathy Miller

Richard Sullivan
IBM Software Solutions Division
Route 100, MD-1335
Somers NY 10589

George Sweeney
IBM
see Cathy Miller

Charles Sweet
4976 Hidden Valley Lane
Parma OH 44129-664

Rob Veitch
Watcom
415 Phillip Street
Waterloo ONT N2L 3X2

Rick Weaver
IBM
see Cathy Miller

Guy Wright
Miller Freeman, Inc.
600 Harrison Street
San Francisco, CA 94107

Peter Yanker
IBM
see Cathy Miller

LAN 0109

LAN Server User Tips and Techniques

Gary Hunt

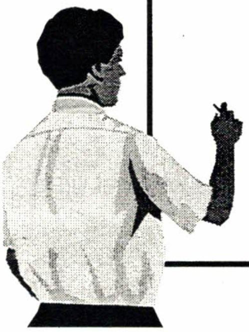
IBM

LAN SERVER 4.0 USER TIPS

LAN SYSTEMS DEVELOPMENT
IBM AUSTIN

LAN SERVER USER TIPS

TOPICS



- USER INTERFACES
- GETTING HELP
- KEY LAN SERVER CONCEPTS
- ACCESSING RESOURCES
- LOGON ASSIGNMENTS
- NETWORK PRINTERS
- SUMMARY

IBM ***LAN Systems***

LAN SERVER 4.0 USER INTERFACES

CHOICES...

NEW!

Graphical User Interface

for OS/2, Windows and DOS

Command Line Interface

User Profile Management (UPM)

NEW!
REXX

Application Programming Interface

IBM *LAN Systems*

GETTING HELP

SOURCES...

- GUI HELP SELECTORS
- COMMAND LINE INTERFACE
  - "NET HELP"
- BOOKS
  - AVAILABLE ONLINE AND IN HARDCOPY
  - EASY START
  - LAN SERVER: UP AND RUNNING!
  - OS/2 LAN REQUESTER USER'S GUIDE
  - DOS LAN SERVICE'S AND WINDOWS USER'S GUIDE
  - LAN SERVER COMMANDS AND UTILITIES
  - OTHERS

IBM *LAN Systems*

KEY CONCEPTS

WHAT ARE ALIASES?

ALIASES ARE LOCATION INDEPENDENT NICKNAMES FOR LAN SERVER RESOURCES

- PROVIDES SINGLE SYSTEM VIEW TO RESOURCES ON MULTIPLE SERVERS
- RESOLVES TO \\SERVERNAME\NETNAME
- USED FOR DEFINING SHARE CONFIGURATION FOR ALL SERVERS IN A DOMAIN
- USED FOR DEFINING USER LOGON CONNECTIONS
- SIMPLIFIES BROWSE AND CONNECT
  - NET USE X: \\SERVER1\APPLET (STANDARD)
  - NET USE X: APPLET (USING ALIAS)

IBM LAN Systems

KEY CONCEPTS

WHAT ARE DOMAINS?

DOMAINS ARE LOGICAL GROUPINGS OF SERVERS

- DOMAIN CONTROLLER
  - PRIMARY AND BACKUP
  - DOMAIN CONTROL DATABASE
  - AUTHENTICATION
  - LOGON ASSIGNMENTS AND PUBLIC APPLICATIONS
- SECONDARY SERVERS
  - LOCAL DATABASE (HPFS386 AND NET.ACC)
  - LOCAL AUTHENTICATION
- MULTIPLE DOMAINS PROVIDE FLEXIBILITY FOR LARGE NETWORKS
  - GREATER CAPACITY AND PERFORMANCE
  - SIMPLIFICATION OF USER VIEW WHILE STILL ALLOWING CROSS DOMAIN ACCESS

IBM LAN Systems

ACCESSING RESOURCES

FOUR STEPS...

1) SHARING A RESOURCE (ADMINISTRATOR)
2) LOGGING ON
   - REGISTERS A USER ID AND PASSWORD IN THE LOCAL CLIENT FOR USE IN SUBSEQUENT RESOURCE REQUESTS
   - GUI, UPM OR COMMAND LINE ("LOGON")
3) CONNECTING TO A RESOURCE
   - ALLOWS A CLIENT TO ESTABLISH A NETBIOS SESSION WITH THE SERVER
   - GRANTS ACCESS TO NETWORK, NOT TO RESOURCES
   - LOGON ASSIGNMENT, GUI SELECTION OR COMMAND LINE ("NET USE")
4) ACCESSING A RESOURCE
   - CONTROL IS ENFORCED WHEN AN ATTEMPT IS MADE TO OPEN A SHARED RESOURCE (EX. "DIR X:\APPS")

IBM *LAN Systems*

LOGGING ON

WHAT HAPPENS WHEN YOU LOGON?

- USER ID AND PASSWORD IS REGISTERED ON LOCAL CLIENT AND USER PROFILE MANAGEMENT

  Tip: Use NETPW to coordinate USERID/PASSWORDS so you will only need to logon once.
- IF MULTILOGON=NO, SINGLE LOGON IS ENFORCED

  Tip: Setting MULTILOGON to YES provides faster logon and the ability to simultaneously logon via multiple workstations.
- NETBIOS CLIENT AND MESSAGING NAMES ARE ADDED

  Tip: Disable the messaging service if you do not use it for faster logon.
- IF LOGON=LOCAL OR DOMAIN, LOCAL SERVER OR DOMAIN CONTROLLER VALIDATES YOUR PASSWORD

  Tip: If not using logon assignments, local validation can be used to bypass Domain Controller and speed up logon.
- DOMAIN CONTROLLER ESTABLISHES YOUR USER LOGON

TION ASSIGNMENTS

IBM *LAN Systems*

FINDING WHAT YOU NEED

NETWORK SURFING...

TECHNIQUES

COMMAND LINE INTERFACE
- NET VIEW - LISTS AVAILABLE SERVERS
- NET VIEW \\SERVERNAME - LISTS RESOURCES ON A SERVER
- NET ALIAS - LISTS ALIASED RESOURCES
- NET USE- LISTS RESOURCES TO WHICH YOU ARE ALREADY CONNECTED

GRAPHICAL USER INTERFACE
- CLICK ON LAN SERVER RESOURCE BROWSER IN NETWORK FOLDER
- CLICK ON APPLICATIONS FOLDER TO VIEW PUBLIC APPLICATIONS

TIPS

USE THE OTHDOMAINS PARAMETER IN IBMLAN.INI TO INCLUDE RESOURCES FROM EXTERNAL DOMAINS YOU COMMONLY ACCESS IN THE RESOURCE BROWSER AND NET VIEW LISTING.

IBM *LAN Systems*

CONNECTING TO RESOURCES

HOW MANY WAYS CAN YOU CONNECT TO A RESOURCE?

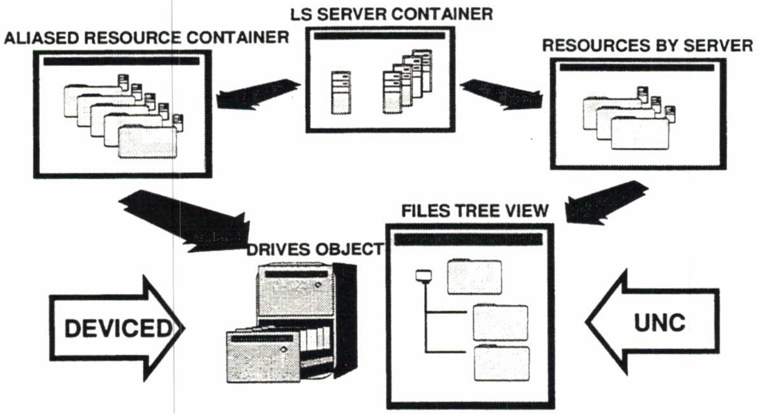
TECHNIQUES

1. COMMAND LINE INTERFACE
   - BY SERVER EXAMPLE: NET USE X: \\SERVER\NETNAME\FILE.EXT
   - BY ALIAS EXAMPLE: NET USE X: ALIASNAME
2. GRAPHICAL USER INTERFACE
   - SELECT OR "DRAG & DROP" ON NETWORK OBJECT
3. LOGON RESOURCE ASSIGNMENTS
   - PUBLIC APPLICATIONS AND ASSIGNMENTS
   - PRIVATE APPLICATIONS AND ASSIGNMENTS
4. DIRECT ACCESS VIA UNIVERSAL NAMING CONVENTION
   - EXAMPLE: COPY \\SERVER\NETNAME\FILE.EXT
5. REXX LAN COMMANDS AND UTILITIES
6. APPLICATION PROGRAMMING INTERFACE
   - NetUseAdd
   - DosOpen UNCName
   - OS/2 GENERIC LAN APIs

IBM *LAN Systems*

CONNECTING TO RESOURCES

NAVIGATION AND CONNECTION WITH OS/2


LS SERVER CONTAINER
ALIASED RESOURCE CONTAINER
RESOURCES BY SERVER
FILES TREE VIEW
DRIVES OBJECT
DEVICED
UNC

IBM *LAN Systems*

CONNECTING TO RESOURCES

USING THE OS/2 WORKPLACE SHELL...

TECHNIQUE

NAVIGATE:

1) NAVIGATE USING THE NETWORK OBJECT TO A RESOURCE

2) ASSIGN A LOCAL LOGICAL ID TO THE REMOTE RESOURCE

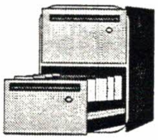
3) CONTINUE NAVIGATION TO THE OBJECT VIA THE NETWORK OBJECT

4) CONTINUE NAVIGATION TO THE OBJECT VIA THE FILES FOLDER

THEN

CONFIGURE:

5) SHADOW THE REMOTE OBJECT INTO A FOLDER ON YOUR DESKTOP



IBM *LAN Systems*

CONNECTING TO RESOURCES

CONSIDERATIONS WHEN USING THE OS/2 WORKPLACE SHELL

OBJECTS SHADOWED FROM SERVERS ARE CLIENT SPECIFIC
- THEY REMAIN WITH A MACHINE AND DO NOT MOVE WITH A USER'S LOGON

OBJECTS DEFINED WITH A UNC NAME ARE DETERMINISTIC
- THEY WILL POINT TO THE SAME SERVER REGARDLESS OF LOGON DOMAIN

OBJECTS MAY ALSO BE DEFINED WITH REDIRECTED LOCAL IDs

TIPS

ASK YOUR ADMINISTRATOR TO CONFIGURE ALL DOMAINS TO HAVE A COMMON SERVICES SERVER TO ALLOW ALL USERS TO HAVE THIS RESOURCE AS A LOGON ASSIGNMENT TO THE SAME DRIVE.

YOUR ADMINISTRATOR CAN ALSO SETUP PUBLIC APPLICATIONS FOR AUTOMATIC ACCESS WHEN YOU LOGON.

LOGON ASSIGNMENTS

TO AUTOMATICALLY CONNECT WHEN YOU LOGON ...

TERMINOLOGY
- LOGON ASSIGNMENTS
- PUBLIC APPLICATIONS
- PRIVATE APPLICATIONS

VIA GUI
- ADMINISTRATOR ADDS RESOURCES TO PUBLIC APPLICATIONS FOLDER
- USER OR ADMINISTRATOR SELECTS RESOURCES FROM LIST TO CONNECT DURING LOGON

VIA COMMAND LINE
- NET APP

SETTING UP A NETWORK PRINTER...

IS EASIER THAN YOU THINK

FROM THE OS/2 DESKTOP

- **OPEN OS/2 TEMPLATES FOLDER**
- **SELECT NETWORK TEMPLATES**
- **FILL IN FORM WITH PRINTER NETWORK NAME, ETC.**
- **YOU WILL BE PROMPTED FOR THE OS/2 PRINTER DEVICE DISKETTE IF THE CORRECT PRINT DRIVER IS NOT ALREADY INSTALLED**

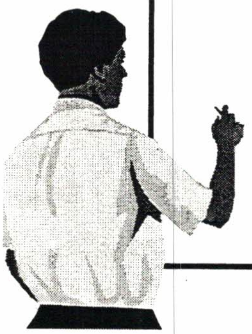
AFTER INSTALLATION, A NETWORK PRINT ICON WILL APPEAR ON YOUR DESKTOP

- **SELECT ICON TO CONNECT**
- **DRAG FILE FROM DRIVES FOLDER AND DROP ON NETWORK PRINT ICON TO PRINT**

IBM ***LAN Systems***

LAN SERVER USER TIPS

SUMMARY



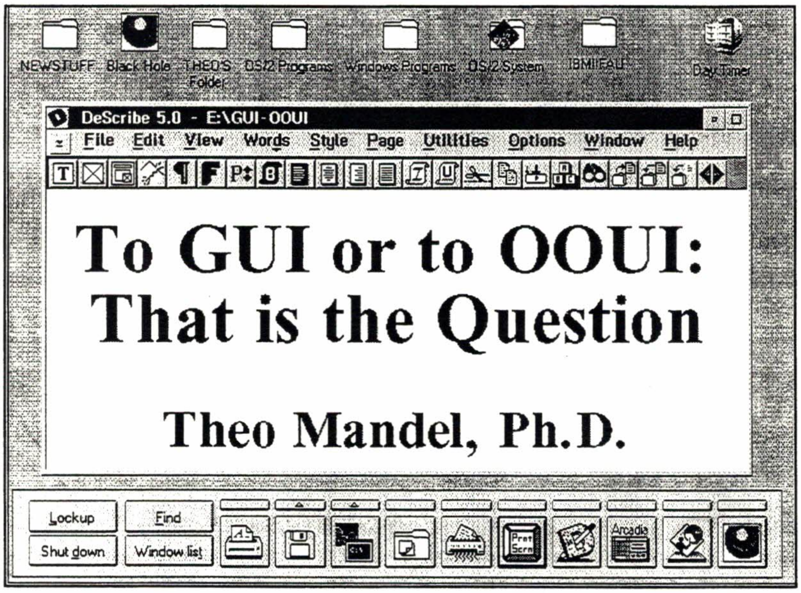
- USER INTERFACES
- GETTING HELP
- KEY LAN SERVER CONCEPTS
- ACCESSING RESOURCES
- LOGON ASSIGNMENTS
- NETWORK PRINTERS

IBM ***LAN Systems***

MGT 0111

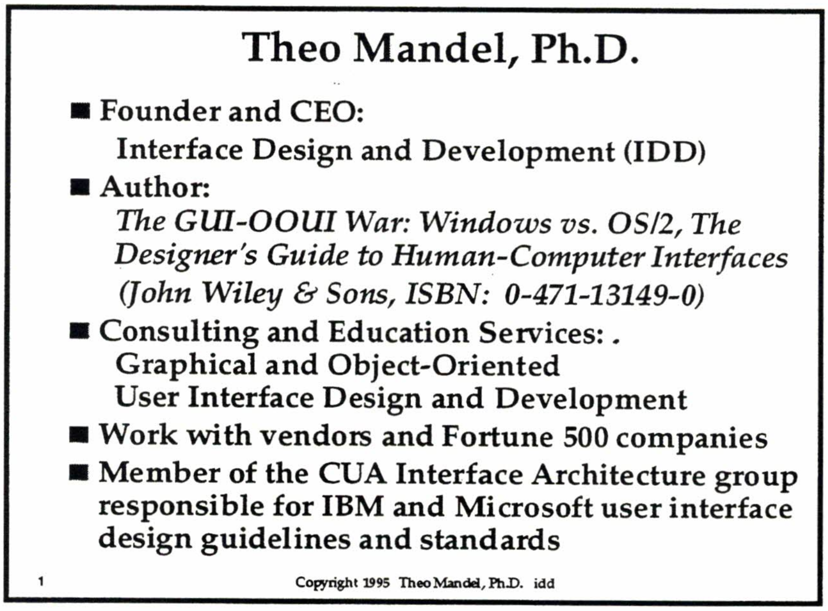
To GUI or to OOUI: That is the Question

Theo Mandel
Interface Design and Development
Austin, TX


NEWSTUFF
Black Hole
THEO'S Folder
OS/2 Programs
Windows Programs
OS/2 System
IBMIFALI
Day-Timer
DeScribe 5.0 - E:\GUI-OOUI
File
Edit
View
Words
Style
Page
Utilities
Options
Window
Help
Lockup
Find
Shut down
Window list
Prnt Scrn
Arcadia

To GUI or to OOUI: That is the Question

Theo Mandel, Ph.D.



Theo Mandel, Ph.D.

- Founder and CEO:
  Interface Design and Development (IDD)
- Author:
  The GUI-OOUI War: Windows vs. OS/2, The Designer's Guide to Human-Computer Interfaces (John Wiley & Sons, ISBN: 0-471-13149-0)
- Consulting and Education Services:
  Graphical and Object-Oriented
  User Interface Design and Development
- Work with vendors and Fortune 500 companies
- Member of the CUA Interface Architecture group responsible for IBM and Microsoft user interface design guidelines and standards

1

Copyright 1995 Theo Mandel, Ph.D. idd

For Further Information, Please Contact:

Theo Mandel, Ph.D.
Interface Design and Development
8001 Two Coves Drive
Austin, Texas 78730-3125

(512) 345-2259 Phone/Fax

CompuServe: 75143,2160
Internet: 75143.2160@compuserve.com

Copyright 1995 Theo Mandel, Ph.D. idd

What to Expect from this Presentation

- Where GUIs and OOUIs came from
- What computer user interfaces are
- How user interfaces affect you and your business, as both developers and users

Awareness | Understanding | Excitement

Copyright 1995 Theo Mandel, Ph.D. idd

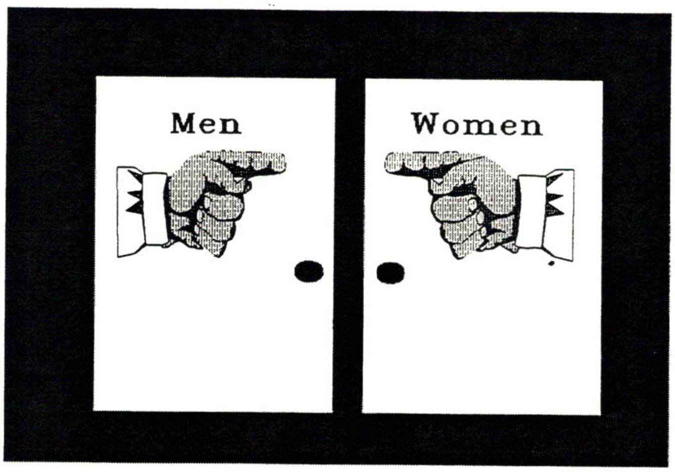
What is an Interface?

- Webster's Dictionary:

The place at which independent systems meet and act on or communicate with each other

Copyright 1995 Theo Mandel, Ph.D. idd

Real-World Interfaces



Copyright 1995 Theo Mandel, Ph.D. idd

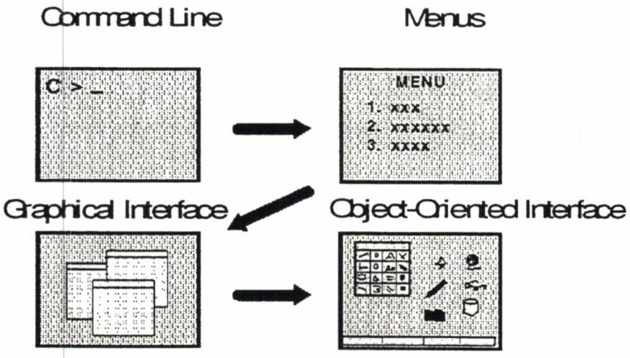
The Human-Computer Interface

- Hardware
  - Display
  - Switches, Buttons
  - Keyboard
  - Mouse
- Software
  - Presentation
  - Interaction
- Product Documentation
- Sensory Characteristics
  - Sound, Voice, Music
  - Animation, Video
- Mental Models
  - User's Conceptual Model
  - Designer's Model
  - Developer's Model

It's not just "Look and Feel"!

Copyright 1995 Theo Mandel, Ph.D. idd

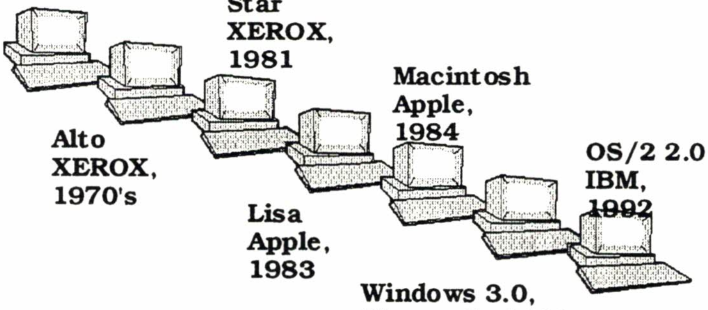
User Interface Styles


Command Line
Menus
C > _
MENU
1. xxx
2. xxxxxx
3. xxxx
Graphical Interface
Object-Oriented Interface

There isn't one perfect style for all users!

Copyright 1995 Theo Mandel, Ph.D. idd

GUI-OOUI Evolution



SketchPad
MIT, 1962

Alto
XEROX,
1970's

Star
XEROX,
1981

Lisa
Apple,
1983

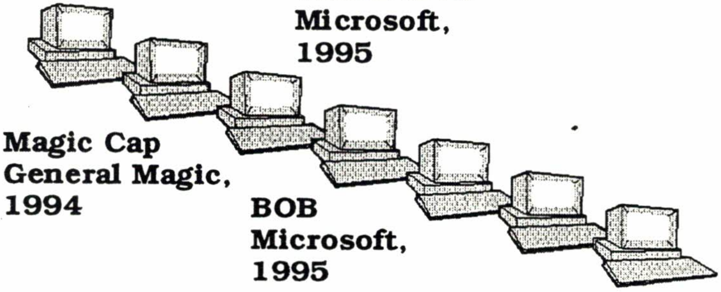
Macintosh
Apple,
1984

Windows 3.0,
Microsoft, 1991

OS/2 2.0
IBM,
1992

Copyright 1995 Theo Mandel, Ph.D. idd

GUI-OOUI Evolution



OS/2 Warp
IBM, 1994

Magic Cap
General Magic,
1994

Windows 95
Microsoft,
1995

BOB
Microsoft,
1995

Copyright 1995 Theo Mandel, Ph.D. idd

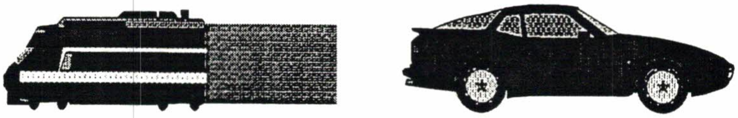
GUI-OOUI Research and Guidelines

- Xerox PARC (Palo Alto Research Center)
- Apple: *Macintosh Human Interface Guidelines*
- IBM Common User Access (CUA)
  - CUA '87 (GUI: IBM/Microsoft)
  - CUA '89 (GUI: IBM/Microsoft)
    - *Workplace Extension described*
  - CUA '91 (OOUI)
  - CUA '95 (OOUI: draft)
- Microsoft Windows
  - *Windows Application Design Guide* (GUI: 1992)
  - Windows User Interface Design Guide (GUI-OOUI: Windows 95)

Copyright 1995 Theo Mandel, Ph.D. idd

Traditional Interfaces vs. GUI-OOUIs

Traditional Interfaces	GUI-OOUI Interfaces
System-driven	User-driven
Focus on processing	Focus on user interaction
Sequential code execution (programmer controls flow)	Event-driven code execution (user controls flow)



Copyright 1995 Theo Mandel, Ph.D. idd

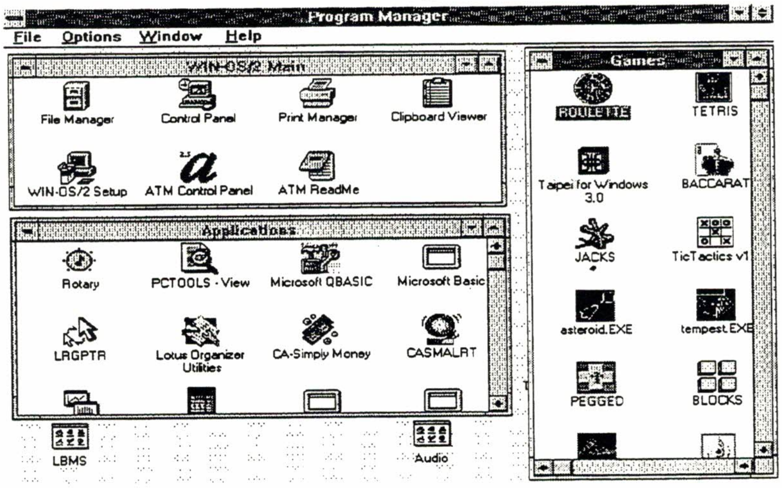
What is a GUI?

- GUI = Graphical User Interface
- WYSIWYG = What You See Is What You Get
- WIMP = Windows, Icons, Mouse, and Pointer

- Visual display of applications and data
- Icons represent applications and actions
- Users must start applications first
- Provides functions to perform user tasks
- Related tasks are supported by other applications
- Training is centered on applications
- Users must follow application structure
- Overlap of functions across applications

Copyright 1995 Theo Mandel, Ph.D. idd

The GUI Desktop: Windows 3.1


Program Manager
File
Options
Window
Help
WIN-OS/2 Main
File Manager
Control Panel
Print Manager
Clipboard Viewer
WIN-OS/2 Setup
ATM Control Panel
ATM ReadMe
Applications
Rotary
PCTOOLS - View
Microsoft QBASIC
Microsoft Basic
LRGPTR
Lotus Organizer Utilities
CA-Simply Money
CASMALRT
LBMS
Audio
Games
ROULETTE
TETRIS
Taipei for Windows 3.0
BACCARAT
JACKS
TicTactics v1
asteroid.EXE
tempest.EXE
PEGGED
BLOCKS

Copyright 1995 Theo Mandel, Ph.D. idd

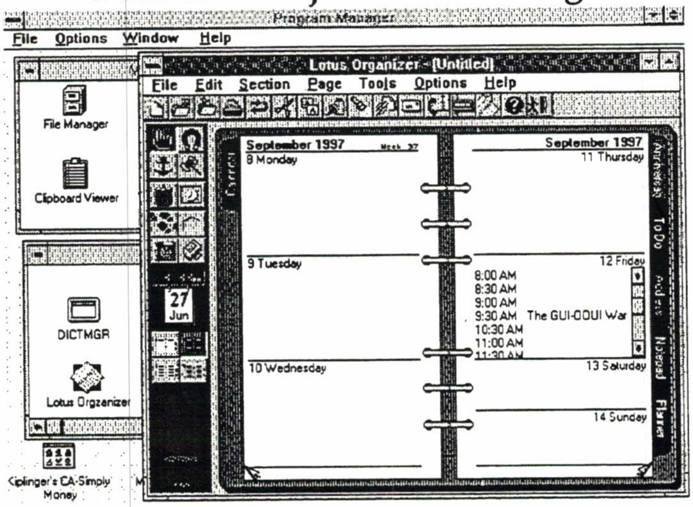
GUIs are Evolving toward OOUIs

- There are degrees of GUI- and OOUI-ness
- Most well-designed GUIs, though still application-oriented, allow users to work with objects within the application

- For example:
- Word Processors and Spreadsheets: text, data, cells, paragraphs, and documents are worked with as objects. Text, graphics, video are all part of composite objects.
- Oher Programs: Personal Information Managers (PIMs) provide users with objects that relate to real-world objects.

Copyright 1995 Theo Mandel, Ph.D. idd

GUIs and Objects: Lotus Organizer


Program Manager
File Options Window Help
File Manager
Clipboard Viewer
DICTMGR
Lotus Organizer
Kiplinger's CA-Simply Money
Lotus Organizer - [Untitled]
File Edit Section Page Tools Options Help
27 Jun
September 1997
Week 37
8 Monday
9 Tuesday
10 Wednesday
September 1997
11 Thursday
12 Friday
13 Saturday
14 Sunday
Calendar
Address
To Do
Notepad
Planner

8:00 AM
8:30 AM
9:00 AM
9:30 AM The GUI-OOUI War
10:30 AM
11:00 AM
11:30 AM

Copyright 1995 Theo Mandel, Ph.D. idd

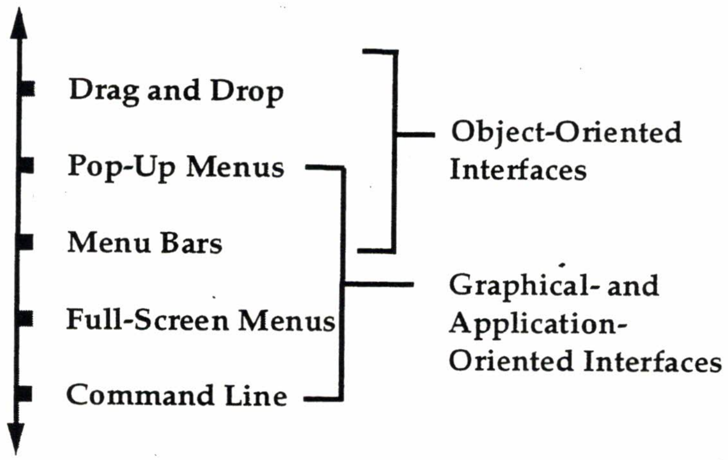
What is an OOUI?

- OOUI = Object-Oriented User Interface
- All of the characteristics of GUIs plus more...

- Products consist of collections of cooperating objects and views, rather than applications
- Users open/work with objects, not applications
- Users more directly manipulate icons and objects
- Objects follow real-world metaphors and models
- Provides supplies and forms to perform user tasks
- Related tasks are supported by other objects
- Training focuses on common objects/techniques
- Users may perform tasks in their own ways

Copyright 1995 Theo Mandel, Ph.D. idd

Object Manipulation Continuum



- Drag and Drop
- Pop-Up Menus
- Menu Bars
- Full-Screen Menus
- Command Line

Object-Oriented Interfaces

Graphical- and Application-Oriented Interfaces

Copyright 1995 Theo Mandel, Ph.D. idd

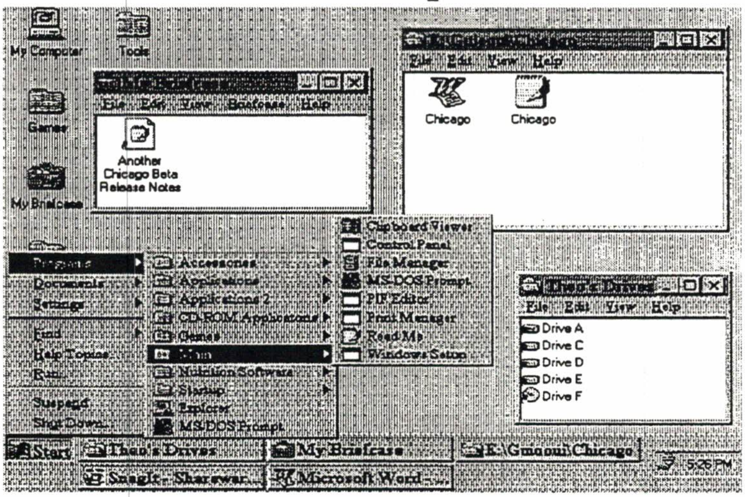
Microsoft's GUI-OOUI Evolution

- Microsoft Windows User Interface Design Guide Preliminary Draft 4.00, May 1994

"The enhancements in the Windows interface pave the way for the evolution from basic graphical user interface design to a more object-oriented user interface design; that is, a more data-centric, rather than application-centric, interface...This is particularly important since this means from the user's perspective that applications become the engines behind the objects in the interface, enabling user interaction, but as transparent to users' tasks as the code that manages menus. This means that while applications will still be important, they no longer are the primary focus of the user. Users should be able to interact with their data without having to think about starting an application, allowing them to better concentrate on their tasks."

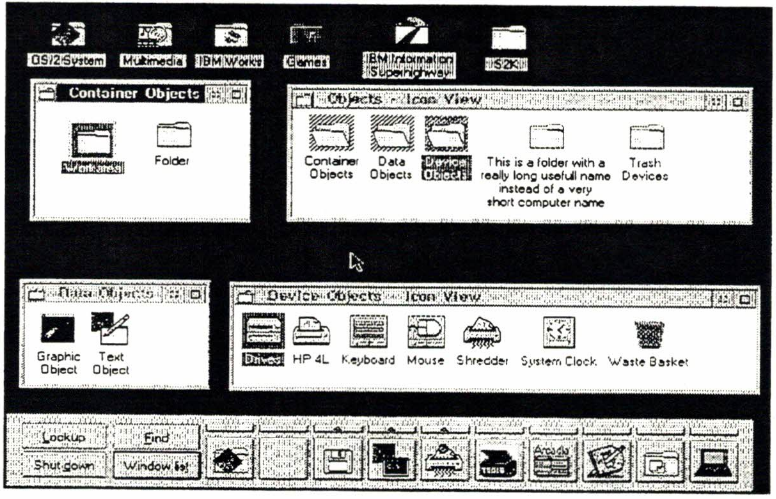
Copyright 1995 Theo Mandel, Ph.D. idd

GUI-OOUI Desktop: Windows 95


My Computer
Tools
Games
My Briefcase
File Edit View Briefcase Help
Another Chicago Beta Release Notes
File Edit View Help
Chicago
Chicago
Programs
Documents
Settings
Find
Help Topics
Run
Suspend
Shut Down...
Accessories
Applications
Applications 2
CD-ROM Applications
Games
Main
Nutrition Software
StartUp
Explorer
MS-DOS Prompt
Clipboard Viewer
Control Panel
File Manager
MS-DOS Prompt
PIF Editor
Print Manager
Read Me
Windows Setup
Theo's Drives
File Edit View Help
Drive A
Drive C
Drive D
Drive E
Drive F
Start
Theo's Drives
My Briefcase
E:\Gmooui\Chicago
Snagit - Shareware...
Microsoft Word - ...
5:26 PM

Copyright 1995 Theo Mandel, Ph.D. idd

OOUI Desktop: OS/2 Warp

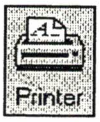

OS/2 System
Multimedia
IBM Works
Games
IBM Information Superhighway
S2K
Container Objects
Folder
Objects - Icon View
Container Objects
Data Objects
Device Objects
This is a folder with a really long usefull name instead of a very short computer name
Trash Devices
Data Objects
Graphic Object
Text Object
Device Objects - Icon View
Drives
HP 4L
Keyboard
Mouse
Shredder
System Clock
Waste Basket
Lockup
Find
Shut down
Window list

Copyright 1995 Theo Mandel, Ph.D. idd

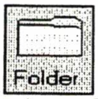
Interface Object Types

- Operating Systems provide predefined objects that users need in their general computing tasks
- These objects follow the desktop metaphors used by the operating system
- Users and developers can modify these object to build more task- and product-specific objects

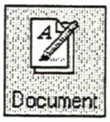
Three types of objects:



Device Objects



Container objects



Data objects

Copyright 1995 Theo Mandel, Ph.D. idd

Views of Objects: Meeting User Needs

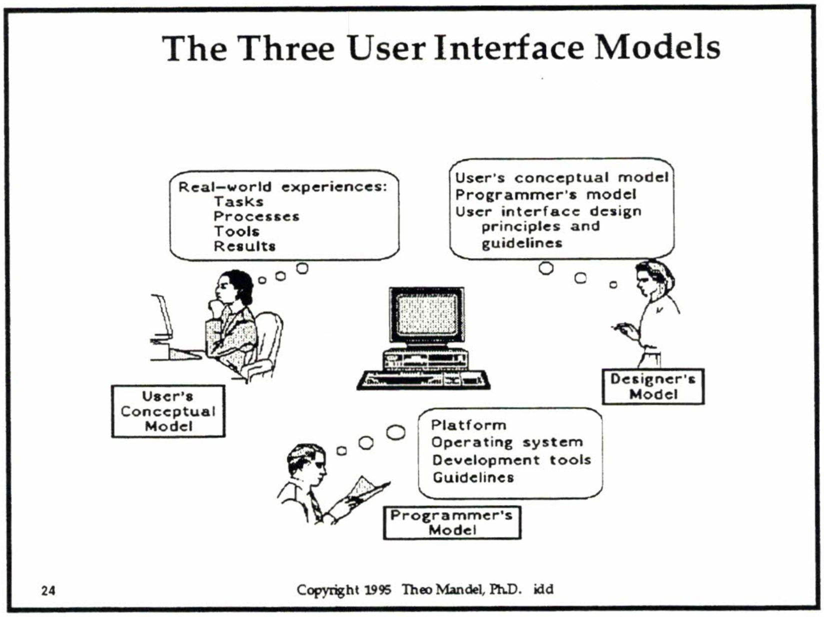
- FIRST: Take great care to determine WHAT objects users need
- SECOND: You must also determine HOW users need (and want) to work with the objects
  - What are the appropriate views?
  - What users need what views?
  - What views are used when?
  - What media should be used?
  - What business process and security measures affect objects and views?
  - Views should be user and context sensitive

Copyright 1995 Theo Mandel, Ph.D. idd

OOUIs and OOP

- OOP = Object-Oriented Programming
- OOUIs and OOP are assumed to always coexist:
  - If you have one then you have the other
  - NOT TRUE!
- Interface objects don't necessarily parallel programming objects
- OOUIs don't (formally) require Object-Oriented Programming (but it helps!)
- OOP doesn't automatically provide usable OOUIs
- OOUIs and OOP complement each other, but they are very different!

Copyright 1995 Theo Mandel, Ph.D. idd



The Three User Interface Models

Real–world experiences:
Tasks
Processes
Tools
Results

User's conceptual model
Programmer's model
User interface design
principles and
guidelines

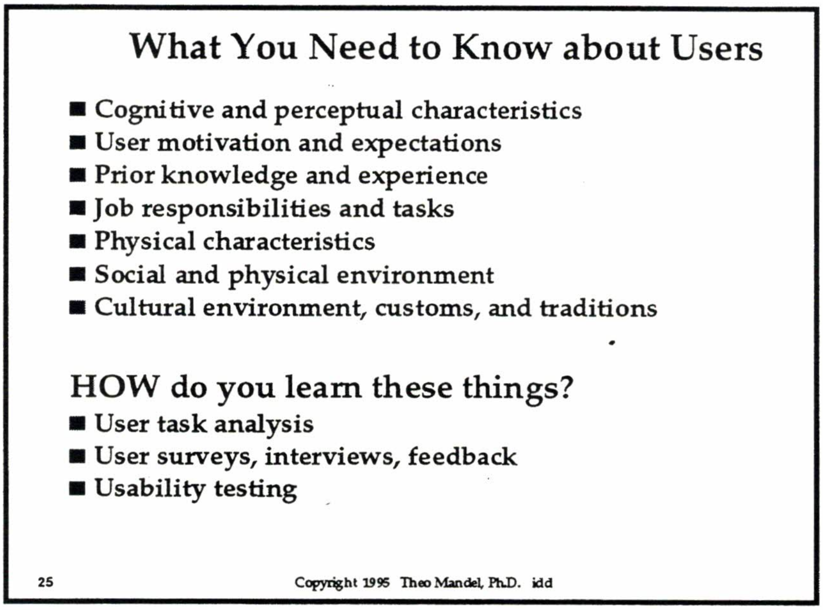
User's Conceptual Model

Designer's Model

Platform
Operating system
Development tools
Guidelines

Programmer's Model

Copyright 1995 Theo Mandel, Ph.D. idd



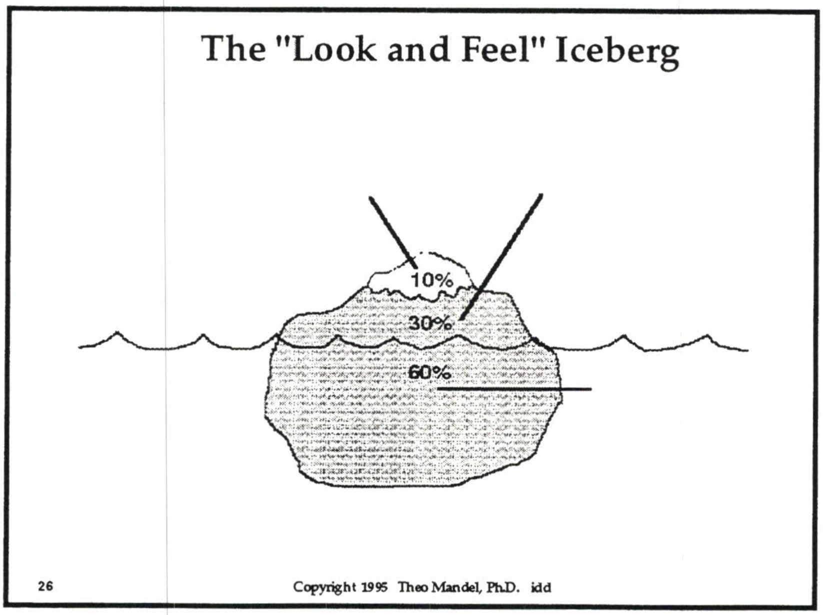
What You Need to Know about Users

- Cognitive and perceptual characteristics
- User motivation and expectations
- Prior knowledge and experience
- Job responsibilities and tasks
- Physical characteristics
- Social and physical environment
- Cultural environment, customs, and traditions

HOW do you learn these things?

- User task analysis
- User surveys, interviews, feedback
- Usability testing

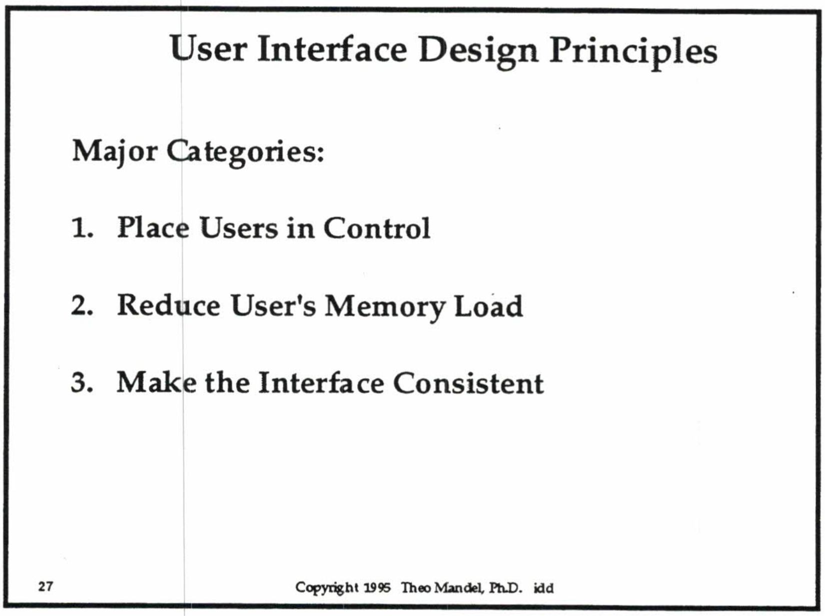
Copyright 1995 Theo Mandel, Ph.D. idd



The "Look and Feel" Iceberg

26

Copyright 1995 Theo Mandel, Ph.D. idd



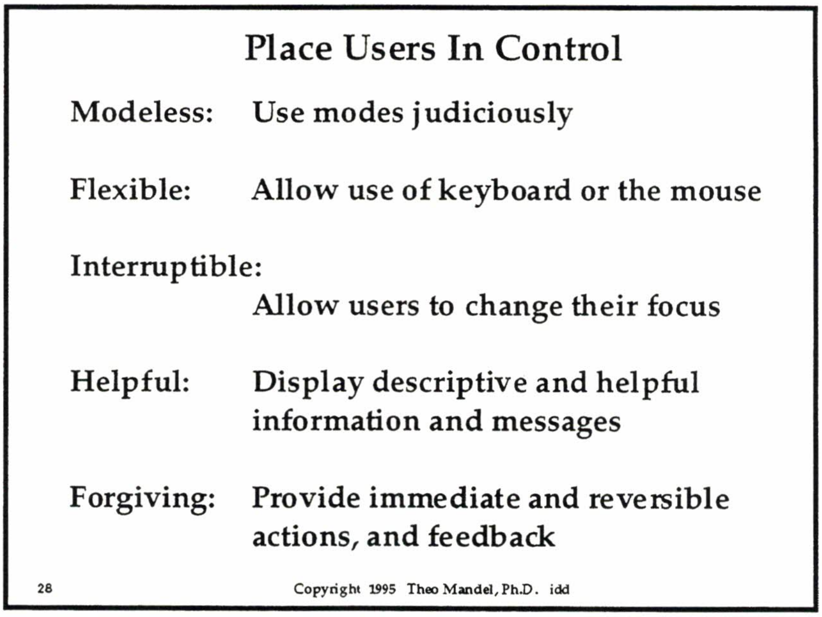
User Interface Design Principles

Major Categories:

1. Place Users in Control
2. Reduce User's Memory Load
3. Make the Interface Consistent

27

Copyright 1995 Theo Mandel, Ph.D. idd



Place Users In Control

Modeless: Use modes judiciously

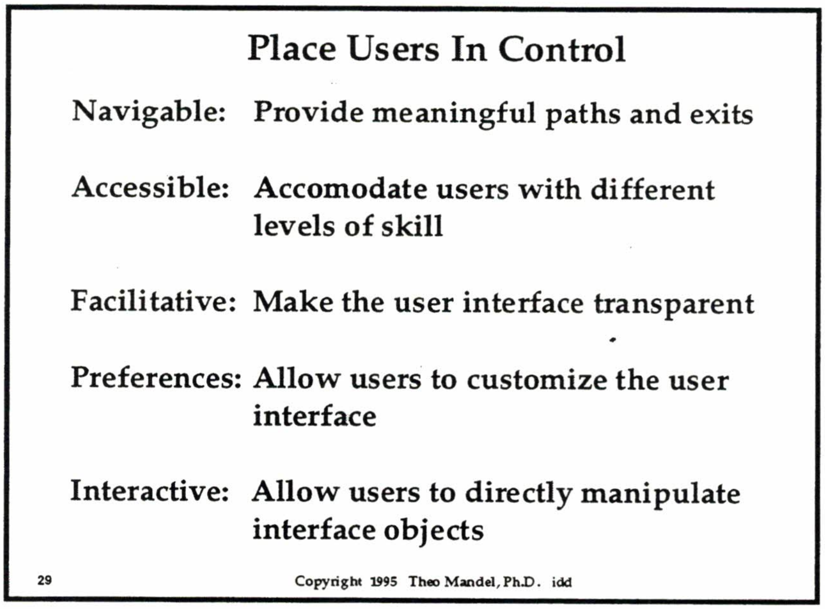
Flexible: Allow use of keyboard or the mouse

Interruptible: Allow users to change their focus

Helpful: Display descriptive and helpful information and messages

Forgiving: Provide immediate and reversible actions, and feedback

Copyright 1995 Theo Mandel, Ph.D. idd



Place Users In Control

Navigable: Provide meaningful paths and exits

Accessible: Accomodate users with different levels of skill

Facilitative: Make the user interface transparent

Preferences: Allow users to customize the user interface

Interactive: Allow users to directly manipulate interface objects

Copyright 1995 Theo Mandel, Ph.D. idd

Reduce User's Memory Load

"Objects should look like they work
and work like they look."

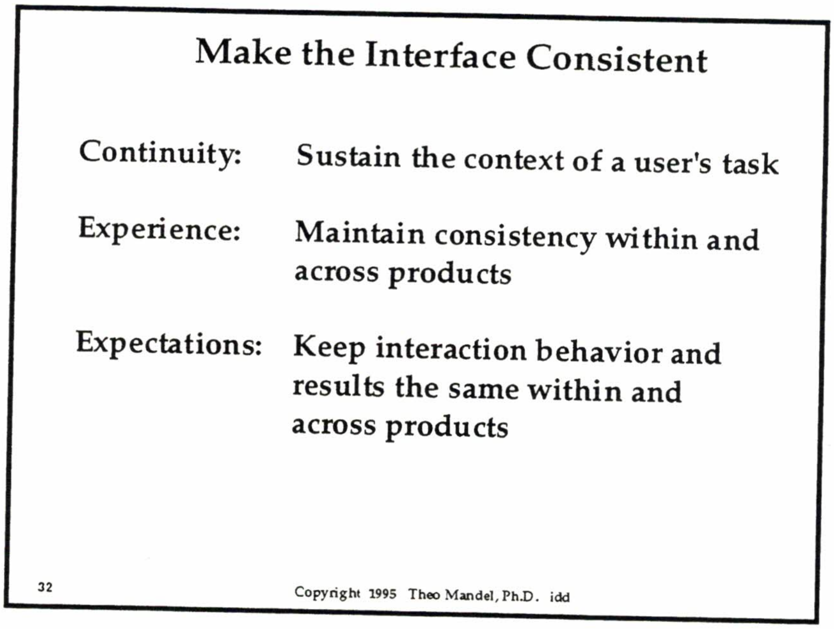
Recall:	Relieve short-term memory
Recognition:	Assist long-term memory
Closure:	Help users complete their tasks
Inform:	Provide visual cues

Copyright 1995 Theo Mandel, Ph.D. idd

Reduce Users' Memory Load

Frequency:	Provide interface shortcuts and accelerators
Aesthetics:	Organize information on the screen
WYSIWYG:	Show what things will look like when users are done
Forgiving:	Provide defaults, undo, and redo capability
Intuitive:	Use an object-action sequence

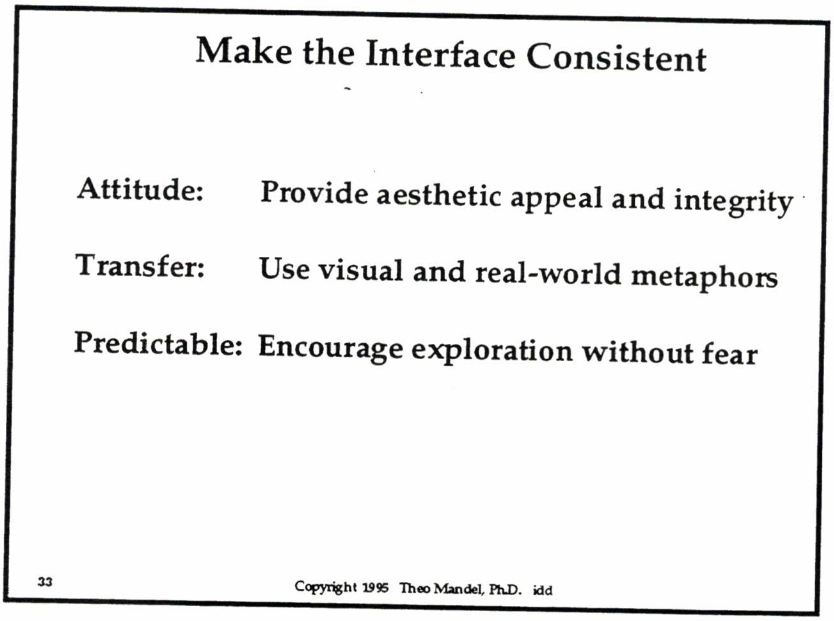
Copyright 1995 Theo Mandel, Ph.D. idd



Make the Interface Consistent

Continuity:	Sustain the context of a user's task
Experience:	Maintain consistency within and across products
Expectations:	Keep interaction behavior and results the same within and across products

Copyright 1995 Theo Mandel, Ph.D. idd



Make the Interface Consistent

Attitude:	Provide aesthetic appeal and integrity
Transfer:	Use visual and real-world metaphors
Predictable:	Encourage exploration without fear

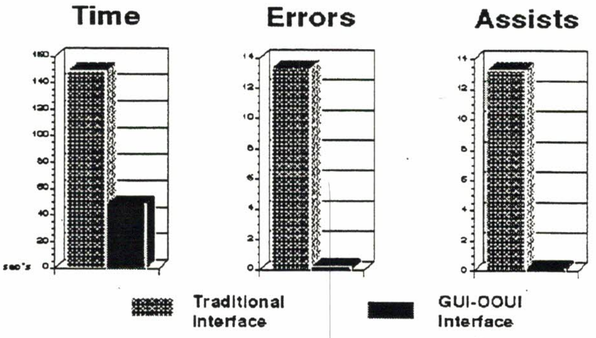
Copyright 1995 Theo Mandel, Ph.D. idd

Benefits and Costs of GUIs and OOUIs

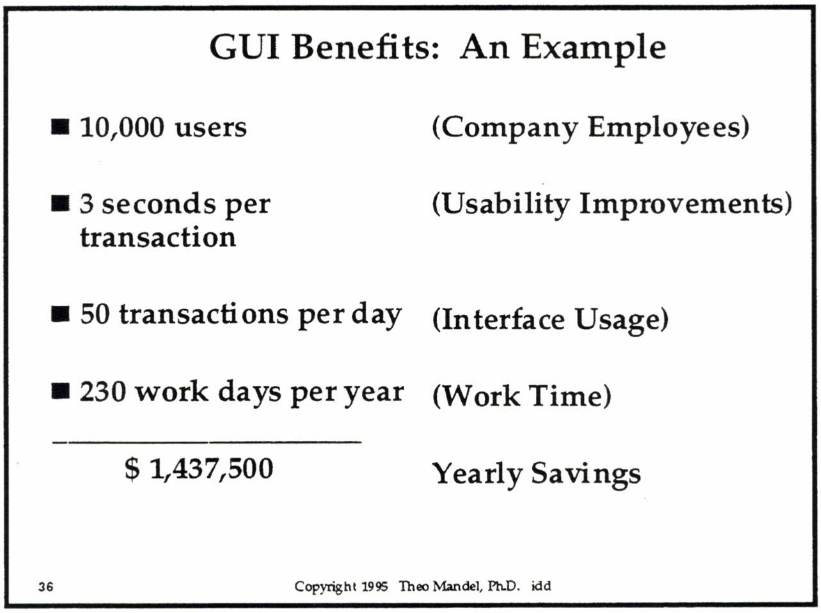
idd

Copyright 1995 Theo Mandel, Ph.D. idd

Example Usability Test Results



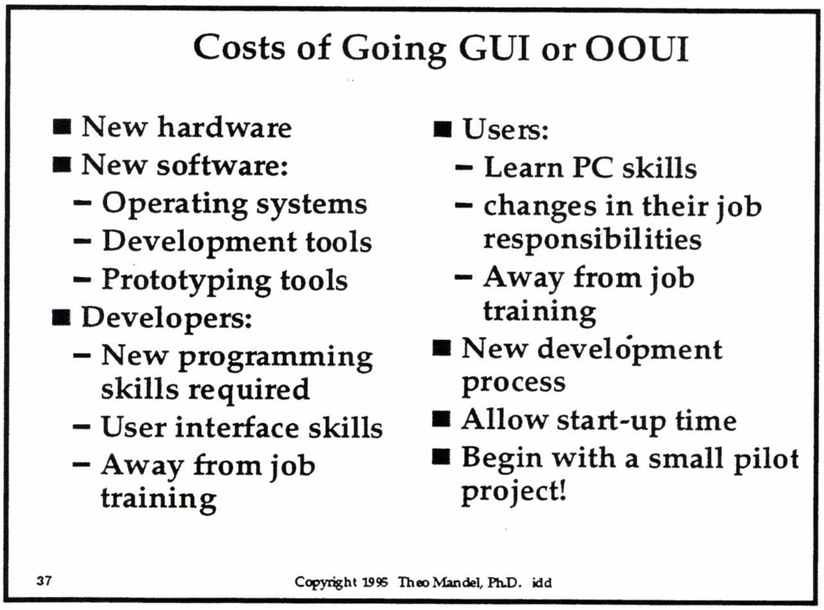
Copyright 1995 Theo Mandel, Ph.D. idd



GUI Benefits: An Example

■ 10,000 users	(Company Employees)
■ 3 seconds per transaction	(Usability Improvements)
■ 50 transactions per day	(Interface Usage)
■ 230 work days per year	(Work Time)
$ 1,437,500	Yearly Savings

Copyright 1995 Theo Mandel, Ph.D. idd

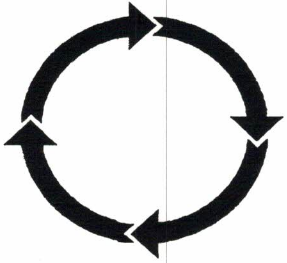


Costs of Going GUI or OOUI

- New hardware
- New software:
  - Operating systems
  - Development tools
  - Prototyping tools
- Developers:
  - New programming skills required
  - User interface skills
  - Away from job training
- Users:
  - Learn PC skills
  - changes in their job responsibilities
  - Away from job training
- New development process
- Allow start-up time
- Begin with a small pilot project!

Copyright 1995 Theo Mandel, Ph.D. idd

The Iterative Development Process



Test

Plan

Design

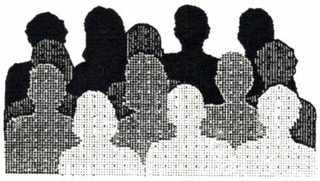
Build, Prototype

Copyright 1995 Theo Mandel, Ph.D. idd

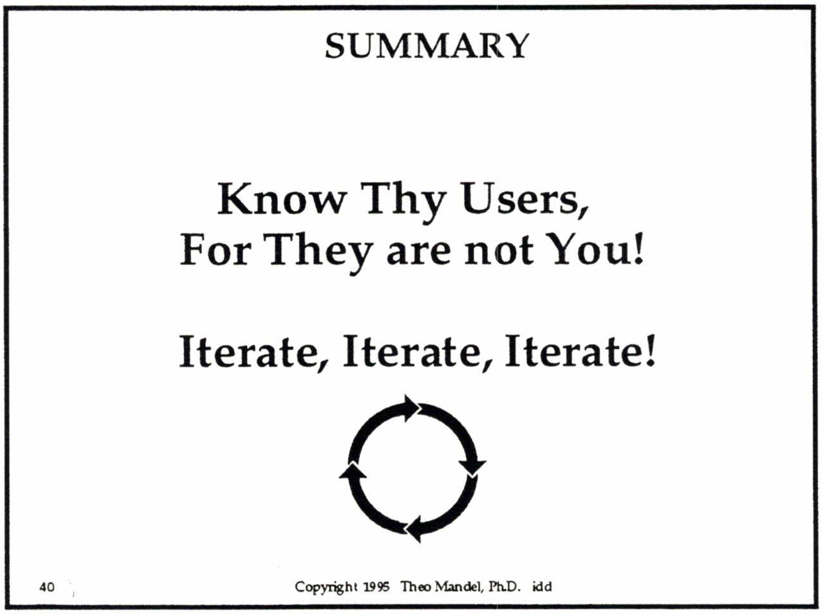
Design Team Approach

Design Team Skills required:

- Programming
- User Interface Design
- Instructional Design
- Graphic Design
- Technical Writing
- Usability Testing
- Business Knowledge Expert
- User



Copyright 1995 Theo Mandel, Ph.D. idd



SUMMARY

Know Thy Users,
For They are not You!

Iterate, Iterate, Iterate!

Copyright 1995 Theo Mandel, Ph.D. idd

MNC 0102

OS/2 Warps the Internet

No paper available. Please contact speaker

this page intentionally left blank

MNC 0110

IBm Personal Systems Service and Support

No paper available. Please contact speaker

this page intentionally left blank

PRO 0101

Designing High Performance OS/2 Applications

David Reich

IBM

Designing High-Powered OS/2 Warp Applications

David. E Reich
IBM Corp
Boca Raton, Fl

Application Features and Initial Specifications

- What's This Thing Gonna Do?
  - It May Seem Overly Simple, But *Decide* on the Primary Function of the Application
  - *Don't* Obscure This Purpose With Fancy Tools or Utilities
  - Utilities Are Just That; Utilities.
  - *Complement* The Main Function. Don't Hide It.

Application Features and Initial Specifications

- How Will You Present the Application?
  - Full Screen?
  - PM?
    - Full PM Windows?
    - AVIO Presentation Space?
  - Workplace Shell Interaction

Application Features and Initial Specifications

- IPC Mechanisms
  - Clipboard
  - DDE
  - Pipes, Queues
  - Data Protocols, Formats and Filters
- REXX Hooks
- Printing
- Fonts

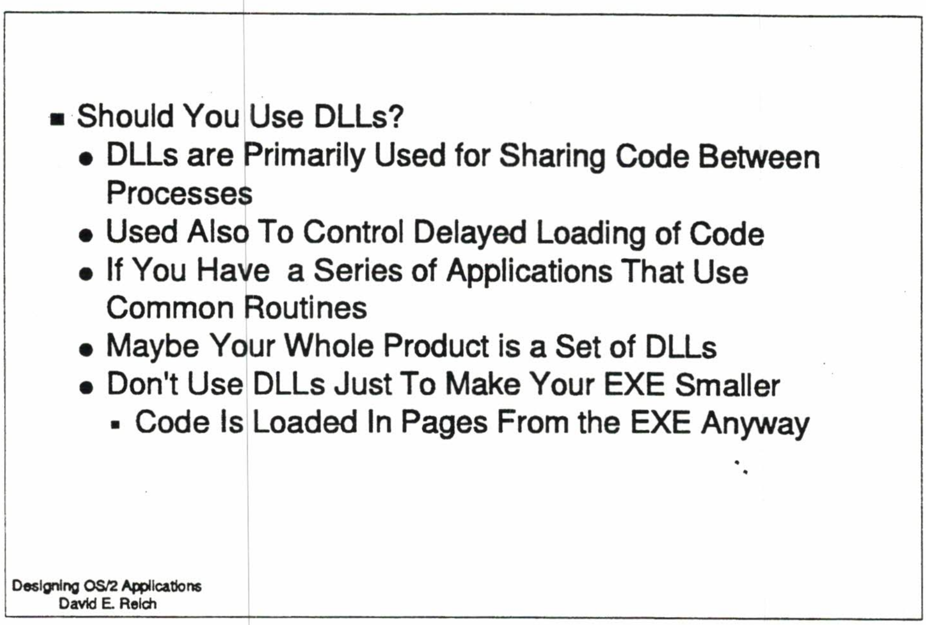
What About the PowerPC?

- OS/2 for the PowerPC is OS/2
- A good OS/2 application is a good OS/2 for the PowerPC application
- Only the plumbing is different
- Based on the IBM Microkernel
    - Originated from CMU's Mach 3
- APIs are the same
- Stay away from assembler, hardware specifics
- Avoid assumptions
    - Query the system for page size, other values
- If you do it right, a port is a recompile

Application Structure

- Isolate From the Underlying Hardware
    - Use the OS/2 APIs and Device Independence
- Use Good Modular Design with Clean Interfaces
- Should You Use Multiple Processes?
    - General rule: Use Multiple Processes When You Need Protection Within the App
    - Communications and Data Sharing Will Be More Involved, However

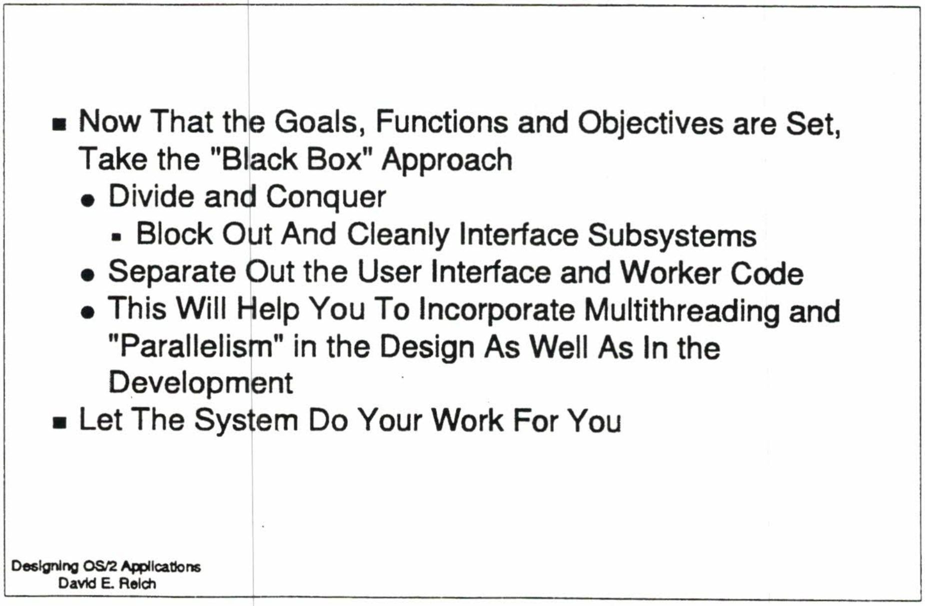
Application Structure



- Should You Use DLLs?
  - DLLs are Primarily Used for Sharing Code Between Processes
  - Used Also To Control Delayed Loading of Code
  - If You Have a Series of Applications That Use Common Routines
  - Maybe Your Whole Product is a Set of DLLs
  - Don't Use DLLs Just To Make Your EXE Smaller
    - Code Is Loaded In Pages From the EXE Anyway

Designing OS/2 Applications
David E. Reich

Building Blocks



- Now That the Goals, Functions and Objectives are Set, Take the "Black Box" Approach
  - Divide and Conquer
    - Block Out And Cleanly Interface Subsystems
  - Separate Out the User Interface and Worker Code
  - This Will Help You To Incorporate Multithreading and "Parallelism" in the Design As Well As In the Development
- Let The System Do Your Work For You

Designing OS/2 Applications
David E. Reich

Designing the User Interface

- Window Design
  - Presenting Data
    - Windows Are Your I/O Medium
  - Window Controls
    - Sizable vs Scalable Data (Scroll Bars?)
    - Menus, Buttons Etc.
  - Dialogs
    - Carry on a Dialog With the User, Then Go Away
    - Must Be Fast, Easy and Intuitive; Flow Smoothly
    - Can Be Used to Control User Interaction With Others
  - Application Defaults
    - USE THEM!

Designing the User Interface

- Workplace Shell Interaction
  - Objects and Templates - Step 1
    - Simple Drag and Drop Already Built In
    - Just Be Able To Take a "Command Line" Parameter
    - User Must Set Up Associations or Use Drives Object For Drag and Drop
  - Objects and Templates - Step 2
    - Create an ASSOCTABLE For The Application
    - When the Shell First Sees It, a Template Will Be Created For Its Documents
    - Custom Icons For Your Document Type
    - Still Just Basic Drag and Drop
    - No User Set Up Required

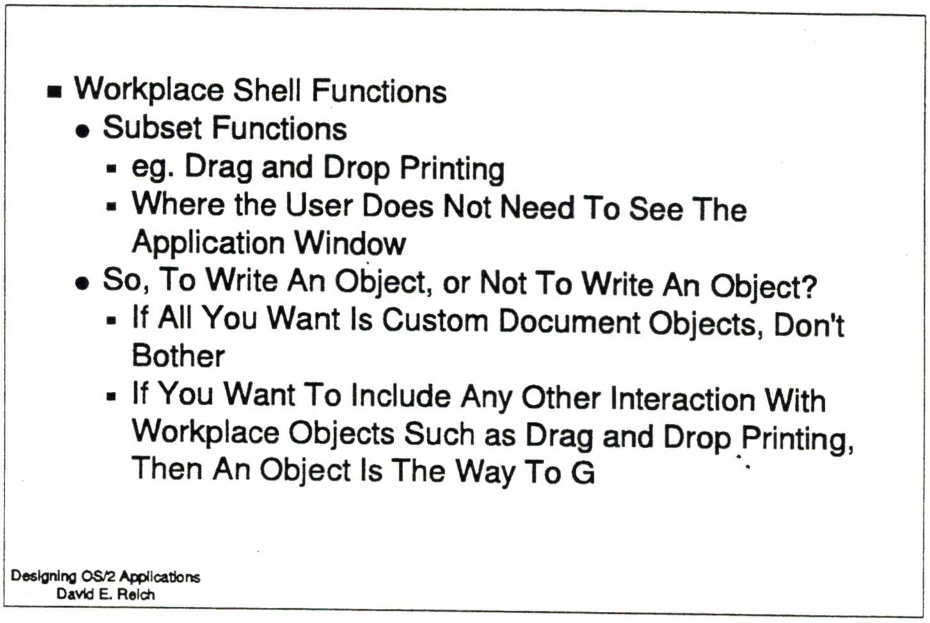
Designing the User Interface

- Workplace Shell Interaction
  - Writing An Object - Step 3
    - Inherits All Previous Advantages
    - Can Perform ALL Functions WIthin the Shell
    - Can Override Methods of Its Class
    - eg. Drag and Drop Printing
    - Dropping onto running instances

Designing the User Interface

- Workplace Shell Functions
  - Application Launching
    - Use Previous Methods
    - DO NOT Write Application As An Object; Workplace Objects Become "part of" the Shell
  - Single Process Model
    - The Shell Is a Single Process
    - Has a "Monitor" Process (The Freddy Shell)
    - Objects Are In DLLs Which Are Run In the Workplace Process
    - With Warp, can now be in separate processes too
    - Communicate With the Launched EXE via DDE etc.

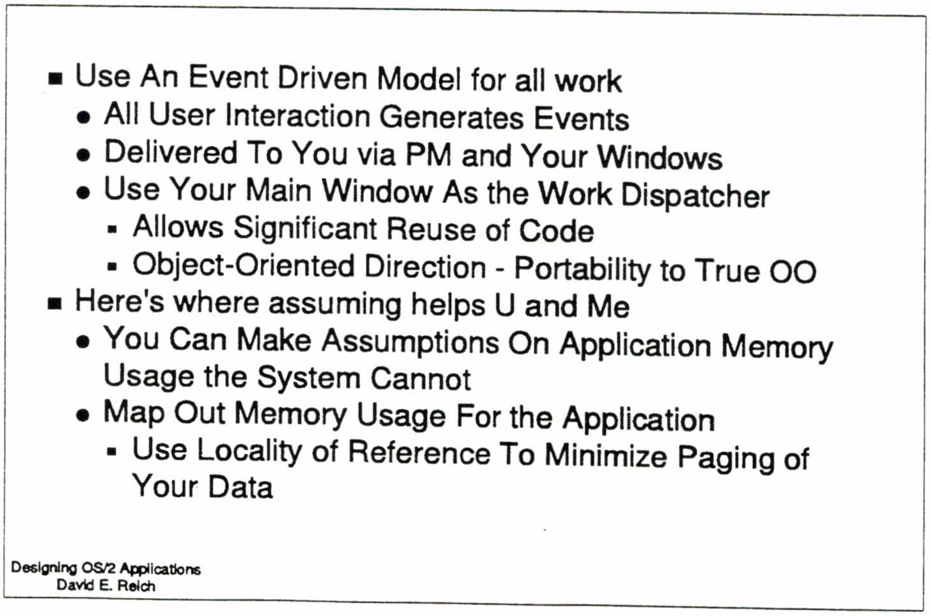
Designing the User Interface



- Workplace Shell Functions
  - Subset Functions
    - eg. Drag and Drop Printing
    - Where the User Does Not Need To See The Application Window
  - So, To Write An Object, or Not To Write An Object?
    - If All You Want Is Custom Document Objects, Don't Bother
    - If You Want To Include Any Other Interaction With Workplace Objects Such as Drag and Drop Printing, Then An Object Is The Way To G

Designing OS/2 Applications
David E. Reich

Designing the Worker Code



- Use An Event Driven Model for all work
  - All User Interaction Generates Events
  - Delivered To You via PM and Your Windows
  - Use Your Main Window As the Work Dispatcher
    - Allows Significant Reuse of Code
    - Object-Oriented Direction - Portability to True OO
- Here's where assuming helps U and Me
  - You Can Make Assumptions On Application Memory Usage the System Cannot
  - Map Out Memory Usage For the Application
    - Use Locality of Reference To Minimize Paging of Your Data

Designing OS/2 Applications
David E. Reich

Designing the Worker Code

- Memory Management (continued)
  - Sparsely Allocate Where Possible
    - Especially Useful For Large, Complex Data Structures
  - Use Guard Pages
    - Register An Exception Handler To Commit When Touched
    - Useful During Development To Fine Tune Usage And Performance - just log the exceptions and analyze
    - If using C/SET, let malloc manage your heap

Designing the Worker Code

- Should I Write a Device Driver?
  - 2 Primary Reasons For a DD
    - If You Need to Write Support For a Specific Piece of Hardware (Scanner, MIDI etc)
    - You Have Work That Must Be Done At Ring 0 and There is NO OS/2 API That Can Do It
  - Isolate Your App From The Hardware
    - Unless Of Course, Your App Is Itself Hardware Support
  - If You Must Code To Hardware Specifics, Write a Generic Interface Package Rather Than Sprinkle DosDevIOCtl Calls All Over Your Code

Designing the Worker Code

- File Layouts
  - EXEs and DLLs
    - Use DLLs For A Purpose
    - Isolate "Subsystems" Such As Hardware Interface Routines in DLLs - They Can Be Replaced Without Rereleasing the Application
    - Controlled In-Memory Code Life
  - INI Files
    - Don't Muck With the System INI FIles
    - Use Your Own INI Files For App-Specific Persistent Information (such as app defaults)

Prototyping the User Interface

- First, Get the Windows Painted
  - Just Get the Windows to Come Up
- Multithreading Considerations
  - Keep Long Jobs In Mind
  - Use the Main Window Proc as a Work Dispatcher
  - Use PM's Thread Handling
    - Object Windows or WinPostQueueMsg
    - WinGetMsg/WinDispatchMsg
    - User Messages (UM_*)
  - Keep the User Input Thread Responsive

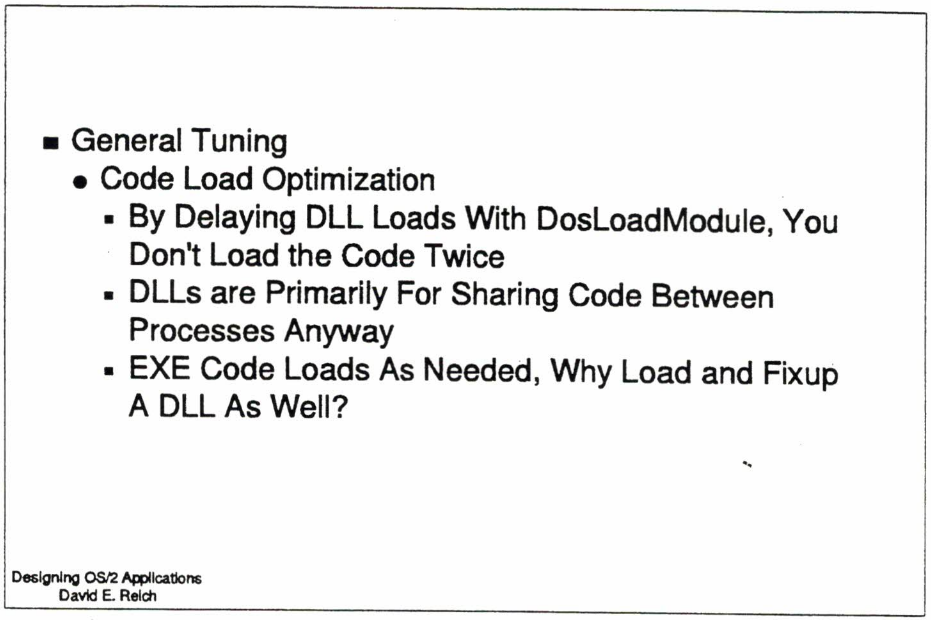
Building the Core Function

- Multithreading and Synchronization
  - Use PM's Object Windows
  - Semaphores
  - Critical Sections (for extreme situations)
  - Should You Even Use Another Thread?
    - Always Think Parallel Operations
    - Make Them Multipurpose (thread pools)
    - Always Consider There is Overhead With Threads
    - Stay Clear of Race Conditions Unless You Are Sure You can Control It and The Performance Gain Is Worth It

Performance

- Memory Tuning
  - Code and Data Working Sets
    - Use Guard Pages To Measure Page Usage
    - Group Cross-referencing Functions In The Same Pages - Locality of Reference
  - Alias APIs
    - For Each Function Not In the EXE, Fixups Are In The EXE Header and In Swappable App Memory
    - Rather Than Have Many Of These, Alias To Your Own APIs, Then You Only Have One Fixup In The EXE
    - Less Paging To Call System Services
    - Speeds Application Load

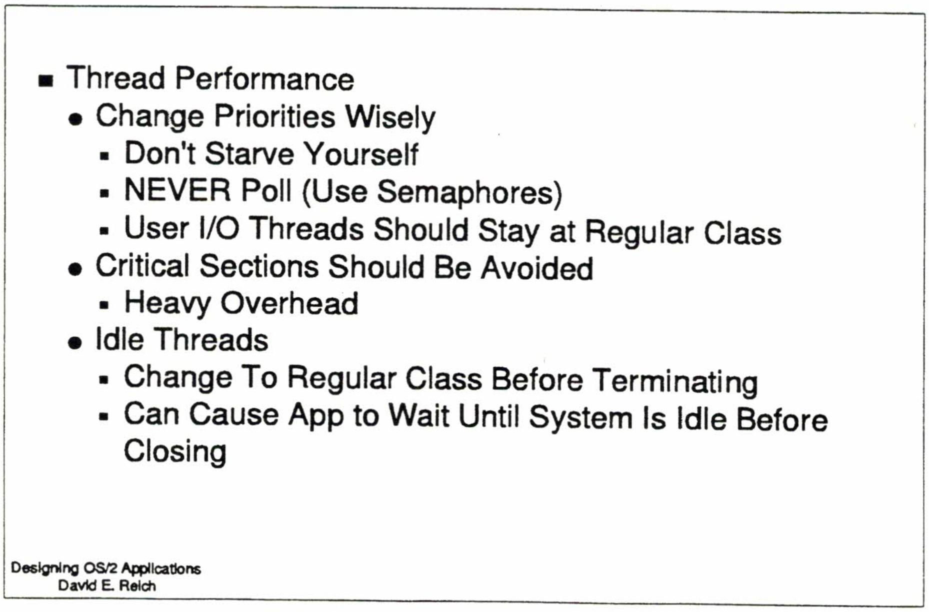
Performance



- General Tuning
  - Code Load Optimization
    - By Delaying DLL Loads With DosLoadModule, You Don't Load the Code Twice
    - DLLs are Primarily For Sharing Code Between Processes Anyway
    - EXE Code Loads As Needed, Why Load and Fixup A DLL As Well?

Designing OS/2 Applications
David E. Reich

Performance



- Thread Performance
  - Change Priorities Wisely
    - Don't Starve Yourself
    - NEVER Poll (Use Semaphores)
    - User I/O Threads Should Stay at Regular Class
  - Critical Sections Should Be Avoided
    - Heavy Overhead
  - Idle Threads
    - Change To Regular Class Before Terminating
    - Can Cause App to Wait Until System Is Idle Before Closing

Designing OS/2 Applications
David E. Reich

Performance

- DLL Placement
  - Don't Fool With User's LIBPATH
    - You Can Hurt Overall System Performance
  - Place Your DLL in Your Own Directory
    - Have Your Installation Program Make Your Directory the Working Directory of the WPS Program Object
    - It Will Be Picked Up Automatically By the "." In the Standard LIBPATH

Performance

- PM Window Performance
  - Try to Keep Often-Used Windows Around
    - WinShowWindow(hWnd, FALSE)
    - They Consume Resources, So Be Frugal
  - Fill Windows Invisibly, Then Show Them
    - Eliminates Flicker
    - Gives Visual Cues to User
  - Let PM Manage Work
    - Don't Recode Things That Default Procs Handle
    - Use Established APIs Rather Than Messages
  - Use Proportions of System-Supplied Values
    - eg. Screen and Printer Page Size

Non-English Language Support

- Stay Away From String Literals
  - Use STRINGTABLES
  - Use Message Files
    - Make Sure To BIND Messages To the EXE or You'll Blow a Whole Page on a 44 Byte Message Segment
- Use Font Metrics and Screen Proportions To Size Windows
  - Will Keep Your App Looking Consistent Regardless of Display Resolution *OR* Language
  - Use WinDrawText (DT_QUERYEXTENT)
- Structure Your Source Trees So All You Need To Do Is Change an Environment Variable To Build Another Language

Designing OS/2 Applications
David E. Reich

Finishing Touches

- The Last Thing Written Is Usually the First Thing Seen -- The Installation Program
  - Don't Neglect It - It IS An Application
  - Make it Look Like the Application
    - Multithread to Make Code Unpacking Faster; Update CONFIG.SYS, Write App INI files
    - Provide Online Help Concurrent With Installation
  - Provide For Multiple Installations on One Machine
    - Creating WPS Objects, Updating System Files etc.
    - Uninstallation - Remove Yourself Completely
  - Make Sure You Write It So You Can Reuse It For A Series Of Applications

Designing OS/2 Applications
David E. Reich

PRO 0104

Enhancing the Enhanced Editor

Bill Schindler
TGC
Cave Creek, Ariz.

Enhancing the Enhanced Editor

Bill Schindler

The Enhanced Editor, or EPM, is included with every copy of OS/2. Although EPM is a full-featured user-configurable editor, most of its power is hidden by a lack of thorough documentation.

Preliminary enhancements

You do some quick, basic enhancements by making some minor setup changes, and changing some of EPM's defaults.

Creating an EPM book

One of the common complaints is that EPM takes a long time to start. You can make EPM start faster by creating an EPM book. The EPM book appears as an icon on your OS/2 desktop.

Setting up an EPM book:

1. Create an EPM program object by copying the Enhanced Editor program object (in the Productivity folder).
2. Open the Settings notebook for the EPM program object.
3. Add "/i" in the Parameters field.
4. If desired, set a default directory (EPM uses this as your default file directory).
5. Select the Window tab and set "Minimize window to desktop."
6. Close the Settings notebook.
7. Shadow the EPM program object to the Startup folder.

EPM starts faster with the EPM book loaded, because the book keeps EPM partially loaded all the time. The EPM book also offers other benefits:

- You can start the editor using drag and drop.
- You can double click on the book to select and edit a file.

Changing the default window size

A common annoyance is that EPM always starts with the same default window size. To force EPM to start with your choice of window size:

1. Resize the EPM window.
2. Select **Options**, then **Save options** from the menu.

If you want the EPM window sized to exact character widths (so it doesn't show cut off characters), use the TRIM command.

Setting up an edit ring

An edit ring allows you to load several files into a single edit window. To turn on the edit ring, select Options, **Preferences**, and then click on **Ring enabled**. Two ring icons will appear on the right side of EPM's title bar. Select **Options**, **Save options** to have EPM always start with the edit ring enabled.

Once the edit ring is enabled, EPM adds some additional functions:

- Add files to the ring by selecting **File**, **Add file** or by pressing F8.
- Remove files from the ring by selecting **File**, **Quit file** or by pressing F3. Quitting from the last file in the ring closes the edit window.
- List and jump to files in the ring by selecting **Options**, **List ring...** or by pressing Ctrl+G.
- Move to the next file in the ring by clicking on the right ring icon (on the title bar) or by pressing F12.
- Move to the previous file in the ring by clicking on the left ring icon (on the title bar) or by pressing F11.

Changing the default font

The default font is usually System Proportional. A proportional font is usually a bad choice for programming, editing many text files, and most other editing (as contrasted to word processing) tasks. Use the EPM menu choice Options, Preferences, Settings... to change the default font. The System Monospaced or the System VIO fonts are usually the best choice.

You can also use this Settings notebook to change the autosave (timed backup) settings, colors, default margins, default tab settings, and the actions performed on Enter key combinations.

Other menu-settable options

Other items on the Options menu can be set according to the your personal preferences. The **Advanced marking** and **CUA accelerators** options are, for the most part, mutually exclusive. You should set these depending on how you prefer to select (i.e. mark), copy, and move text.

Creating a profile

Many of EPM's settings are only accessible via EPM commands. If you need to enter the same commands (using Ctrl+I) at the beginning of each EPM session to make EPM work your way, then you'll want to create a profile.

For example, EPM defaults to automatically expanding program text as you type. I personally find this annoying, so I shut off code expansion by pressing Ctrl+I and entering the command EXPAND OFF. A profile can be used to perform the same function:

```
/* profile.erx for EPM */
'expand off'
```

Save this file as PROFILE.ERX. The profile must be saved in a directory listed in EPMPATH (defined in CONFIG.SYS). For a standard OS/2 setup, the normal directory would be \OS2\APPS.

You need to tell EPM to use the profile. Open the EPM command line by pressing Ctrl+I, enter PROFILE ON, then select **Options**, **Save options**.

Now, whenever you start EPM, code expansion will be turned off. You can use the profile in this manner to customize EPM to a very large extent. For example, here's a profile that changes the cursor to an underline, the tab key to use words in the prior line as tab stops, and the escape key to perform the same function as Ctrl+I (i.e. open the command dialog):

```
/* profile.erx */
'cursor_style 2'
'matchtab on'
'escapekey on'
```

For a list of EPM commands, select **Help**, **Commands help**.

Using EPM REXX macros

The EPM profile is actually an EPM REXX macro. An EPM REXX macro has all of the power of the REXX programming language available to it, along with many extensions supplied by EPM. EPM REXX macros also have access to most system REXX extensions (like RexxUtil) Both Watcom VX-REXX and Hockware VisPro/REXX can generate REXX macros that are PM-based applications.

You can get a collection of sample EPM REXX macros from ftp://software.watson.ibm.com, CompuServe's OS2SUPPORT forum, or the Walnut Creek Hobbes CD-ROM. (I will demonstrate a few of these sample macros during the session.)

An EPM REXX macro is executed from EPM's command line using the command RX *macroname*. If the macro is stored in a directory listed in EPMPATH, you don't need to enter a directory name as part of the macroname.

EPM commands are executed in a REXX macro as shown in the section on creating a profile. EPM provides an addition command, EXTRACT, that is only available to a REXX macro. The EXTRACT command allows the macro to not only retrieve text but to also retrieve the current state of several EPM internal variables.

EPM also provides functions so that a REXX macro can replace or insert lines of text.

Here's an EPM REXX macro that converts all the text in a file to uppercase:

```
/* uppercase.erx */
'extract /last'                 /* get number of last line */
DO i = 1 TO last.1
  i                             /* position to line number */
  'extract /getline'            /* get text of current line */
  line = Translate(getline.1)   /* uppercase the text */
  CALL EtkReplaceText line      /* replace the current line */
END
```

The EPM macro language

The Enhanced Editor has its own macro language called "E". The E source code and the E compiler, ETPM, don't come with OS/2, but they're available on the Hobbes CD-ROM, on the internet at ftp://software.watson.ibm.com and on CompuServe in the OS2SUPPORT forum. If you get a copy of E, also be sure to get a copy of the technical and user online documentation.

Using E, you can modify and customize almost everything the Enhanced Editor does. The E macro language is a programming language in its own right and is too extensive of a topic to cover in this session. The technical and user documentation discuss using and programming in E and in combination with samples included with the source code, should help you quickly come up to speed.

If you want to reconfigure EPM with a minimum amount of programming, you can create a file named MYCNF.E, set several configuration variables, and recompile the macros.

```
; example MYCNF.E
const
  AUTOSAVE_PATH     = 'd:\saved\' -- Put AUTOSAVE files here
  P_SYNTAX_ASSIST   = 0           -- Don't do Pascal, so save space
  SMARTFILE         = 1           -- If no changes, F4 just quits.
  SMARTQUIT         = 1           -- If changes, F3 says "Y, N or File"
  EXTRA_EX          = 1           -- Required for EPM

/* Initializations... */
my_default_save_options='/S'      -- Strip trailing spaces when saving.
```

The macro source includes two sample MYCNF files that will guide you in customizing EPM. Be sure to also read the file READ.ME before making any wholesale changes. Also, see STDCNF.E for all the possible values that you can set in MYCNF.

SYS 0105

How to Create a User-Friendly OS/2 GUI/OOUI

Luca Miccoli
IBM

Abstract: This lecture is aimed at anyone who needs to design and/or develop a Graphical Object Oriented User Interface under OS/2 2.x or OS/2 Warp Version 3. It provides a set of guidelines, hints, and tips for making an OS/2 user interface that is both powerful and easy to use, integrating it in the OS/2 Workplace Shell, and making it exploit the features of the main OS/2 Presentation Manager graphic controls (Notebooks, Containers, Sliders, Value Sets) and OO paradigms (Context menus, Drag and Drop). The lecture presents and analyzes several typical situations and paradigms of object oriented user interfaces, providing solutions and implementation guidelines under OS/2.

1. MAIN DIFFERENCES BETWEEN A GUI (GRAPHICAL USER INTERFACE) AND AN OOUI (OBJECT ORIENTED USER INTERFACE)

When you think of a graphical user interface, the first thing that might come to your mind is the Microsoft Windows 3.X desktop, which is probably the most famous GUI product on the market.

Anyone familiar with this environment or with a GUI environment in general can easily realize that the paradigm used puts the emphasis on the applications. An application is generally represented as an icon, and is started with a mouse interaction (double-click) with that icon.

Icons can be grouped together to form groups associated with specific tasks such as handling network connections, multimedia devices, and games.

GUI applications are typically menu-driven, and rely on action bars and pull-down menus to drive specific actions. They use list boxes to present collections of items such as actions and selectable choices and they provide direct manipulation techniques, such as drag and drop, that sometimes are bound to the single application and do not span across multiple applications. Settings are generally handled through dialog boxes and saved in application proprietary formats.

A typical example of object oriented user interface is the OS/2 WPS (Workplace Shell), introduced with OS/2 2.0. The OOUI paradigm puts the emphasis on objects instead of applications. The OS/2 Desktop is a collection of objects (represented with icons), and the user interacts with them to do specific tasks. Objects are a computer representation of real- life business-objects like a printer, a FAX, and a shredder; the user can interact with them emulating real-life behaviours and situations.

In the OS/2 Desktop applications lose their identities and all the objects contained in an application can be seen as extensions of the Workplace Shell, which creates a seamless integration.

Each object has a pop-up (context) menu that contains all the actions associated with the object; this type of menu allows a manipulation technique complementary to drag and drop, which is the most direct one. Two other important features of the OS/2 Workplace Shell are folders (container objects in general) and notebooks that are used for containing other objects and displaying information about them. In particular, there is a templates folder that contains template objects that can be cloned to create new objects.

In short, the main differences between a GUI and an OOUI are:

- Icons represent applications in a GUI and objects in an OOUI.
- Applications have a well-defined identity and are menu-driven in a GUI, while in an OOUI they are seamlessly integrated with the Desktop and are formed by several objects that can be manipulated directly or by using context menus.
- List boxes are used in a GUI to show collections of items, while containers are used in an OOUI.
- Settings of a GUI application are handled using several dialogs, and are generally stored in application proprietary formats. In an OOUI settings are handled using notebooks, and .INI files are generally used to save them.
- Direct manipulation techniques are generally application-specific in a GUI, while all objects in an OOUI are

manipulated using drag and drop or context menus.
- An OOUI uses a template folder as a factory to create other objects, while a GUI does not have an equivalent concept.

2. WHY IMPLEMENT A GUI UNDER OS/2?

After knowing all the benefits of OOUIs and of the OS/2 Workplace Shell, it would seem obvious that an OS/2 user interface should be object oriented and perfectly integrated in the OS/2 Workplace Shell. In reality, there may be two cases where an OS/2 application cannot have a true OOUI:

1. An OS/2 PM (Presentation Manager) application was originally written for OS/2 1.X and runs under OS/2 2.X or OS/2 Warp without problems. In this case, if the effort of restructuring the application cannot be afforded, there will not be a seamless integration with the OS/2 WorkPlace Shell; however, some actions for improving the integration can be taken (see paragraph 6).

2. An application is designed to be portable through multiple environments and operating systems. This is a typical situation when developing a client/server application that must have a client part with a GUI running on several platforms (for example, DOS, Windows 3.X, OS/2 1.X, OS/2 2.X and Unix). In this case, if the user interface code is "common", only a GUI with a subset of portable features can be implemented.

3. BASIC GUIDELINES AND TIPS FOR WRITING A WELL- BEHAVED OS/2 PM APPLICATION

"OS/2 PM application" is a generic term to identify an OS/2 program with a graphical, windowed interface. Given that the Presentation Manager is the OS/2 window-managing system and the OS/2 Workplace Shell itself is a PM application, the guidelines provided in this paragraph are valid for any OS/2 graphical interface. It must be noted that there are really many suggestions that can be given to write a well- behaved OS/2 PM application; what follows is only a small set of basic guidelines that can apply to the most general situations. The main items represent guidelines, while subitems represent the associated explanations, hints, and tips.

KEEP THE KEYBOARD AND MOUSE INTERACTION HIGHLY RESPONSIVE

- All the user interface specific code should be put in a separate (generally the main) thread, while other time-consuming operations (for example, retrieving data from a database) should be run in other threads.

- In particular, if it is necessary to process a big file, lock it using semaphores (to obtain a safer locking you might even open the file and deny write access to other processes), and do the bulk work in a separate thread. This allows the user to switch to other applications and exploit the multitasking capabilities of OS/2. Think how boring it is to open a big file with a word processor and then wait minutes before being able to operate not only the word processor but the whole PC...

- In terms of performance, it is better to post user messages than to send them. You can use object windows running in separate threads as message routers.

- Do not make an excessive use of processes, sessions, threads, and IPC (Inter Process Communication) mechanisms like shared memory, semaphores, pipes, queues, etc. Sometimes a load-balance analysis of your application must be performed to find a compromise solution.

- Avoid creating semaphores inside a window procedure unless it runs in a dedicated thread, because this can cause a deadlock situation.

- The processing of a message inside a window procedure should not last more than 0.1 seconds.

MANAGE UNPREDICTABLE SITUATIONS SUCH AS UNRECOVERABLE ERRORS

It is possible to write an exception handler routine to manage severe error stituations, like memory access violations, that cause an OS/2 application to terminate abnormally. Useful information about the code instruction that caused the severe error can be provided by an exception handler routine. This is important especially for PM applications, because the default behaviour in the case of unrecoverable error is simply to terminate the program without providing any information about the cause of the error. The IBM C Set++ compiler provides a compiler option (/Tx) that generates a default exception handler that in most cases is quite acceptable.

MAKE WINDOW PROCEDURES REENTRANT USING RESERVED WINDOW WORDS

Avoid using global variables to make window procedures access needed information. When registering a new window class for an application you can reserve storage by specifing a number of bytes as the last parameter of the WinRegisterClass API (Application Programming Interface). When a window of the registered class is created, the specified number of bytes is reserved for it; this memory can be set using the WinSetWindowUlong and/or WinSetWindowPtr APIs , and accessed (inside the window procedure) using the WinQueryWindowUlong and/or WinQueryWindowPtr APIs.

USE .DLL (DYNAMIC LINK LIBRARY) FILES TO SPLIT THE APPLICATION IN SEPARATE AND REUSABLE COMPONENTS TO ALLOW CLEAN INTERFACES

Code reuse is not an exclusive feature of object oriented technology. The code of a procedural PM application can also be designed to be reusable; and DLLs are a powerful way for implementing clean interfaces and modularity. Try to keep the code in a DLL as independent as possible, to avoid cross-dependencies with other application DLLs.

USE APPLICATION .INI FILES AND DO NOT PUT INFORMATION INSIDE SYSTEM .INI FILES

OS2.INI and OS2SYS.INI are really critical files, and should not contain information that could be put in application specific profile (.INI) files. Exploit the Prf APIs to create application specific .INI files where information like window positions and sizes can be stored.

MAKE A PM APPLICATION NLS (NATIONAL LANGUAGE SUPPORT) ENABLED

Resource, message, and help files are the main features OS/2 provides to separate the translatable information from the actual appplication executable code. The NLS resources of a PM application like menus, dialogs, and STRINGTABLEs should be packaged in one or more dedicated .DLL files, while message and help files have their own formats and extensions.

4. MAKING YOUR USER INTERFACE EASY TO USE

It is almost unnecessary to say that user- friendliness and usability are key factors for the success of a software product, especially in the PC world. Object oriented user interfaces have been invented to mirror the real world and let the user feel comfortable when using a PC. The IBM CUA (Common User Access) guidelines are the best source of information for all the usability factors that must be considered when designing and implementing a GUI or an OOUI under OS/2. Sometimes, small usability features make the difference between OS/2 products: this paragraph contains a list of suggestions that if implemented (for some of them this is really straightforward) can give your PM application a superior quality.

- As much as possible use visual representations of objects from the user's world. Let the user work by analogy.

- Let the users always feel that they are controlling the application and not vice versa. Sometimes it can be really frustrating for a user to be driven by the application and always be worried about doing something wrong.

- Provide at the first glance only the information needed by a novice user, but provide also an easy way to retrieve more detailed information for the experienced user.

- Let the user perform similar tasks in the same way (for example, show the same dialog to select two different types of files).

- Never hurt the users' sensibility by using messages that state that they did something wrong.

- Take care when designing the icons of your application not to use symbols or pictures that the user could consider to be offensive.

- Provide suitable defaults for all the customizable operations of your application.

- If any multiple selection mechanism is implemented (for example, in a container or in a list box), always provide a "select all" and "deselect all" action.

- Hide programming implementation details from the users. They should see your application as something helping them to solve problems, and should not care about the particular way it accomplishes this.

- When providing a new version of your application, keep the user interface consistent with the one of the previous version or provide an option to switch back to it. Carefully document changes between versions.

- Prevent exceptions and user errors by not allowing the users to execute incorrect actions (for example, make menu items or drag and drop operations unavailable).

- Customizable tool bars and action buttons give a nice look and feel to a user interface; but avoid cluttering the screen with too many visual objects. For each action button provide an equivalent menu item in the menu bar, and implement "fly over" help.

- Provide contextual on-line help with "search" facilities and hyphertextual links. "How can I do something" information should also be provided or as part of a tutorial program, or contained in the on-line help itself.

- During the execution of an installation program, use a progress indicator to let the user know the state of the installation. Providing short information (small dialogs or bitmaps) on the application while the installation program is running can be very impressive for the user.

5. HINTS AND PIFTALLS OF THE MAIN OS/2 PM CONTROLS AND OOUI PARADIGMS

This paragraph is divided in subsections associated with a specific OS/2 PM control or OOUI paradigm. Each subsection describes typical use situations and gives some hints of things to do and pitfalls to avoid.

5.1 NOTEBOOK

Typical use situations:

- Manage the settings of an object.
- Present data that can be logically organized into groups.

Hints:

- Whenever possible, make notebook pages look similar, to prevent their flickering when turning over them.
- If your application has a big notebook that is displayed very often, it might be more efficient to create it only once and then to display and hide it instead of always creating and destroying it.
- Notebooks tend to occupy a big portion of the screen, so take care when determining their size.

Pitfalls:

- Sometimes information can be diffficult to retrieve because the related notebook tabs are not immediately selectable and, therefore, the user must scroll to display them.

5.2 CONTAINER

Typical use situations:

- Allow a user to group and view objects in ways that are not provided by the system-provided containers.

Hints:

- Use 1 byte alignment for the C structures or C++ classes associated with the records of a container; otherwise memory violation errors might randomly occur in your application.
- Use a tree view container when you want to show hierarchical data in your application. The Tree text view might be not as nice as the Tree icon view, but can perform better and can display more information in the same screen space.
- If it is known that the expansion of a record in a tree view container will take considerable time, it might be wise to ask the user for a confirmation of the operation or give a visual representation of the progress of the situation.

Pitfalls:

- You might experience poor performance when repainting details view containers, especially under OS/2 2.X.
- Specifying a delta to handle a large amount of records in a container is not as useful as it sounds. In fact, the scroll bars in the container are sized and positioned according to the actual number of records inside it, not to the full set of records intended to be viewed.

5.3 SLIDER

Typical use situations:

- See the current value of a measurement relative to the range of possible values.

Hints:

- Horizontal sliders are generally more suitable than the vertical ones. Use vertical sliders when trying to represent objects that resemble a thermometer or the level of a liquid in a container.

Pitfalls:

- When resizing a slider control, the position of the arm is not automatically adjusted, so it is an application's responsibility to correctly reposition the arm.

5.4 VALUE SET

Typical use situations:

- Show mutually exclusive graphical or short textual choices.

Hints:

- When a mutually exclusive color choice has to be implemented, use a value set with VS_COLORINDEX style instead of a group of radio buttons.
- A value set with 1 row and VS_BITMAP style can be used to implement a tool bar.

Pitfalls:

- If a value set has the VS_BITMAP style but does not have the VS_SCALEBITMAPS style, each bitmap is centered in its cell. If the cell is smaller than the bitmap, the bitmap is clipped to the size of the cell.

5.5 CONTEXT MENU

Typical use situations:

- Show a pop-up menu containing choices appropriate for the context of a given object or set of objects in an appplication.

Hints:

- Calculate accurately the position for displaying a context menu, so that it does not cover the associated object.
- If your application allows this, let the user customize a pop-up menu by adding or removing choices; make the customization persistent.

Pitfalls:

- Be sure to destroy a context menu after the selection of a menu item makes it disappear; otherwise the menu will remain unnecessarily loaded in memory.

5.6 DRAG AND DROP

Typical use situations:

- All application objects represented as icons on the OS/2 Workplace Shell must have drag and drop capabilities.
- Implement drag and drop between visual objects when it is an intuitive shortcut to a copy or move operation.

Hints:

- Always disable drag or drop operations when source or target object do not support the operation.
- Provide a way to cancel a drag and drop operation or provide an "undo" capability.

Pitfalls:

Only operations between a group (1 or more) of source objects and a target object can be effectively implemented using drag and drop.

6. INTEGRATING A USER INTERFACE IN THE OS/2 WORKPLACE SHELL

The best way to integrate an application in the OS/2 Workplace Shell is to create it using SOM (System Object Model) to exploit the WPS classes with their hierarchy. This requires some experience with object oriented technology and in particular with SOM. However, as has already been said before, the OS/2 Workplace Shell is just a PM application, and another PM application can have the same look-and-feel and create a sinergy with it. This is possible because all PM applications can use controls like containers and notebooks and can implement the same direct manipulation techniques used to operate WPS objects. The following basic guidelines can be used to better integrate a plain PM application in the Workplace Shell:

- Use containers, making them look like WPS folders.
- Provide context menus for both icons and windows.
- Provide drag and drop capabilities for the application items. Use OS/2 File RMF (Rendering Mechanism and Format) to support dragging to or from WPS folders. Use OS/2 Print RMF to allow printing by direct manipulation.
- Put dialogs related to similar tasks such as application settings together in a notebook.
- Make the application program accept a file name as its first command line parameter, and put an ASSOCTABLE statement in the resource script file of the application program. This will create an association for the program and will allow the creation of new template objects in the Templates folder.

Migrating a plain PM application to a full SOM/WPS one might not be straightforward, but usually parts such as window procedures and functions not specific to the user interface can be reused or easily modified.
The most important considerations to make when planning to develop a full SOM/WPS application are:

The Workplace Shell is a single- process PM application, and all WPS applications (packaged as DLLs) share the resources of this process. Hence, if your WPS application does not make a good usage of threads, the overall performances of the system can be heavily impacted.

You might need to change your attitude to design or develop a user interface that fully exploits the OOUI features and paradigms.

REFERENCES

Object Oriented Interface Design: IBM Common User Access Guidelines (IBM DOC. SC34-4399)
OS/2 Version 2.0 - Volume 3: Presentation Manager and Workplace Shell (IBM DOC. GG24-3732- 00)
OS/2 3.0 Control Program Programming Guide (IBM DOC. G25H-7101)
OS/2 3.0 Presentation Manager Programming Guide - The Basics (IBM DOC. G25H-7103)
OS/2 3.0 Presentation Manager Programming Guide - Advanced Topics (IBM DOC. G25H-7104)
OS/2 3.0 Workplace Shell Programming Guide (IBM DOC. G25H-7108)
The GUI-OOUI War — Windows vs. OS/2: The Designer's Guide to Human-Computer Interfaces (Theo Mandel - 1994 - Van Nostrand Reinhold)

SYS 0106

Communications Manager/2: Overview and Directions

George Sweeney

IBM

COMMUNICATIONS MANAGER/2

Overview & Directions

George Sweeney
Networking Software
Products
Research Triangle
Park, NC

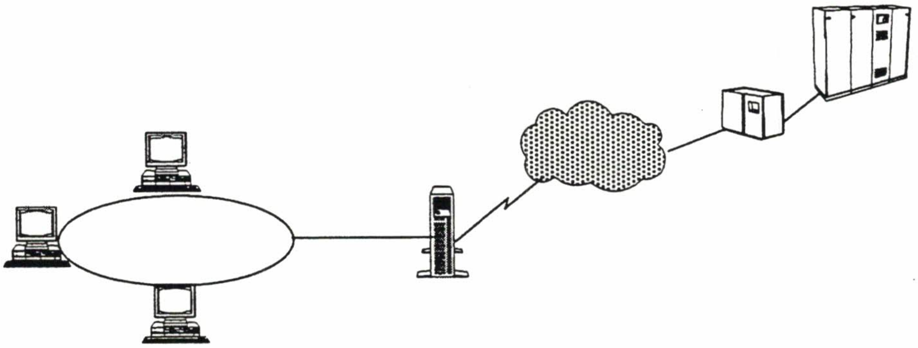
Communications Manager/2

➡ **Today's Product - CM/2 1.11**

► **Emulator Enhancements**

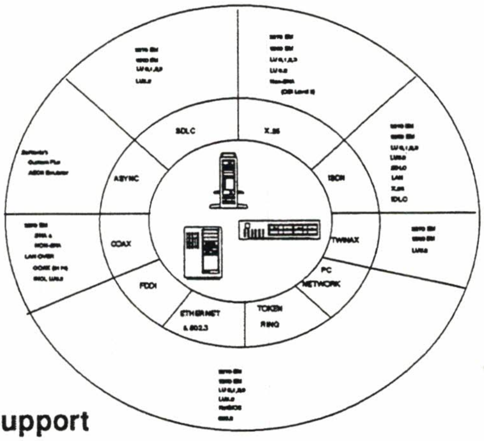
► **Future Directions**

Gateway



- Provides LAN Workstation Access to Host
- Supports up to 254 Logical Units per Physical Connection
- Supports Multiple Host Connections
- Supports Dynamic Addition or Update of Workstations Without Reconfiguring

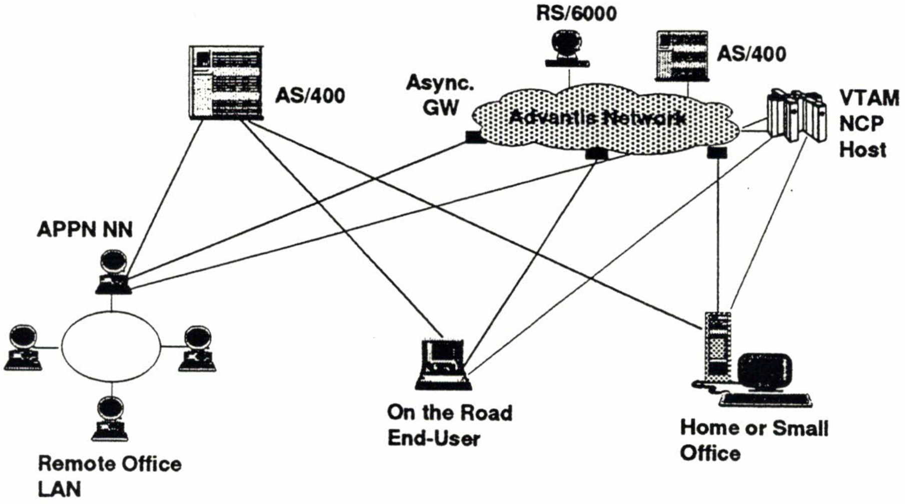
Connectivities


SDLC
X.25
ISDN
TWINAX
PC NETWORK
TOKEN RING
ETHERNET & 802.3
FDDI
COAX
ASYNC

- Data Compression Support
  - Improves Effective Transfer Rate
  - Provides SNA Data Compression at Session Level
  - Uses Run Length Encoding (RLE) and Lemphel-Ziv 9-bit(LZ9)
- Wide Area Connector (WAC) Adapter Support
  - Provides 64 Kbps Capability
  - Supports 2 SDLC Lines
- Additional Adapter Support

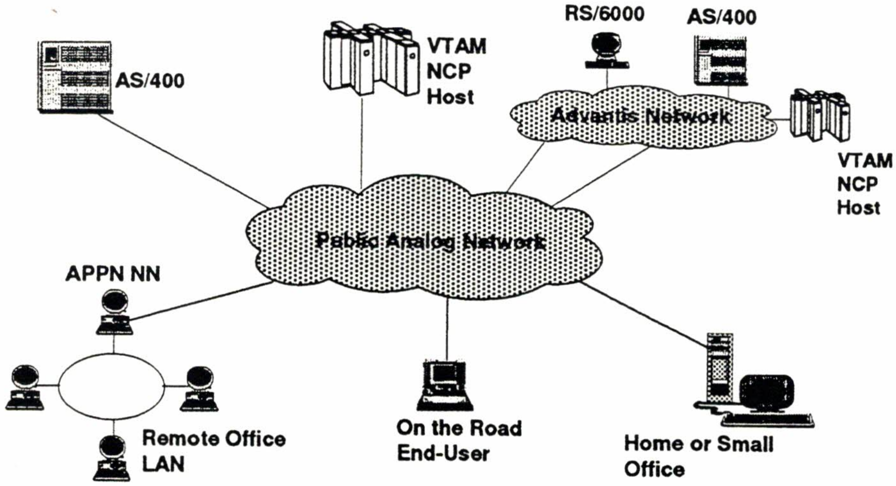
Async SNA Phone Connect Scenarios

- SNA Applications (3270 Em.)
- LU 6.2 Applications (5250 Em.)


RS/6000
AS/400
AS/400
Async. GW
Advantis Network
VTAM
NCP
Host
APPN NN
On the Road End-User
Home or Small Office
Remote Office LAN

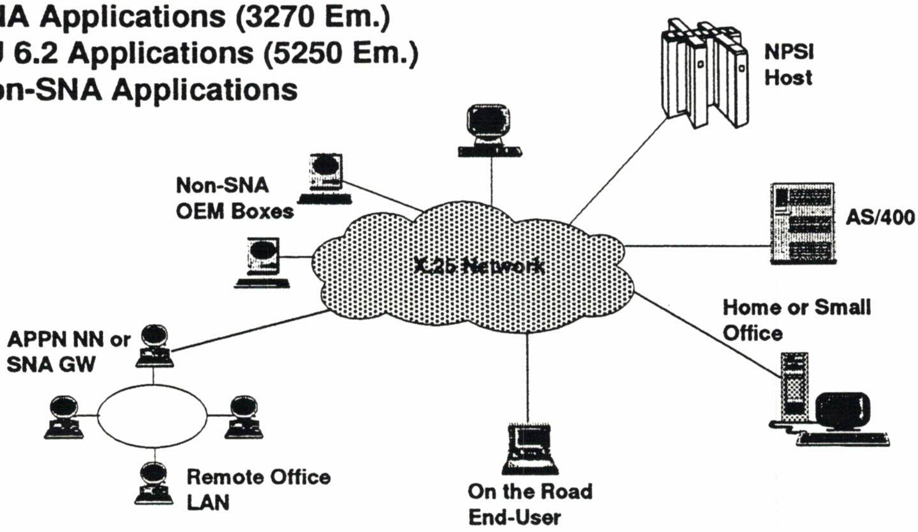
AutoSync or Sync SNA Phone Connect

- SNA Applications (3270 Em.)
- LU 6.2 applications (5250 Em.)


AS/400
VTAM
NCP
Host
RS/6000
AS/400
Advantis Network
VTAM
NCP
Host
Public Analog Network
APPN NN
Remote Office LAN
On the Road End-User
Home or Small Office

AutoSync/Sync Access to X.25 Networks

- Dial-up Access to X.25 Networks
- SNA Applications (3270 Em.)
- LU 6.2 Applications (5250 Em.)
- Non-SNA Applications

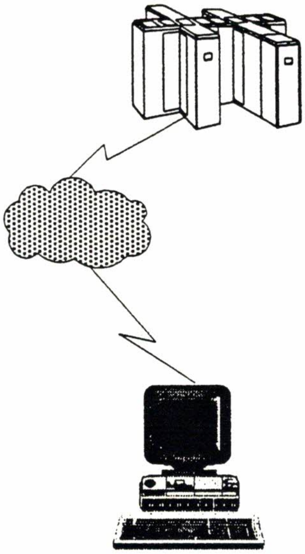

NPSI
Host
Non-SNA
OEM Boxes
X.25 Network
AS/400
Home or Small
Office
APPN NN or
SNA GW
Remote Office
LAN
On the Road
End-User

Programming Interfaces

- System and System Services
  - Communications Manager Kernel
  - Common Services
  - System management
  - SPA Router/ROP Services

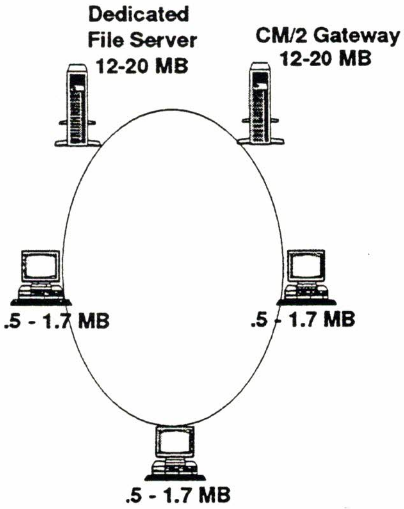
- Communications
  - EHLLAPI
  - SRPI
  - LUA
  - APPC
    - Full Duplex Conversations
    - Non-Blocking Verbs
  - CPI-C Level 2.0
  - ACDI
  - X.25
  - Real-time Interface Co-Processor (RTIC)
  - ISDN

Emulators



- 3270 and 5250
  - Same Look and Feel
  - Dynamic Run Time Options
  - Hot Spots, Pop-Up Keypads
  - Automatic Font Selection
  - Menu Bar
  - Numeric Lock Field
- 3270
  - Session Level Encryption
  - Up to 26 Active Sessions (OS/2 2.1)
  - Menu-Driven File Transfer
  - Response Time Monitor
  - PCMCIA Support
  - Import/Export Support
  - Integrated Host Graphics
  - APL Support
- 5250
  - Text Assist for PC Support
  - PC Organizer

Distributed Feature

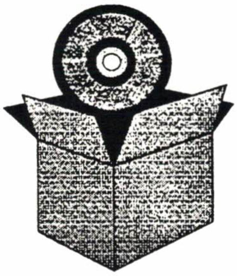


- Significantly Reduce DASD Requirement
- Simplify Installation & Maintenance
- Provide Code Consistency
- Allow Installation of
  - Multiple Service Levels
  - Multiple National Language Support Versions
- Create & Change Configurations at Distributed Feature Workstation

National Language

- Translated Versions
  - French
  - German
  - Italian
  - Japanese
  - Korean (Standard)
  - Spanish
- Double Byte Character Set Version
  - Chinese (Simplified)
  - Chinese (Traditional)
  - Japanese
  - Korean (KS)

Packaging



- CM/2 Product Code
  - CD-ROM Installable (CD-ROM Only)
  - Diskette Images
- CM/2 Applets
- API Support Code (CD-ROM Only)
- NTS/2 Product Code
- SOFTERM Custom Plus, ASCII Emulator
- Installation and Configuration
- Quick Install Card
- User's Guide
- Online Publications
  - Command Reference
  - Messages Reference
  - Overview
  - Glossary
  - Problem Determination Guide
- Publications Order Form

PLUS.........

Packaging: CD-ROM

Communications Manager/2 1.11

- User's Desk Reference
- Installation & Configuration
- Planning & Administration for Networks
- Host Connection Reference
- Host Print Reference
- Scenarios
- Planning & Information Guide
- Application Programming Guide
- APPC Programming Reference
- EHLLAPI Programming Reference
- ISDN CMI Programming Reference
- ISDN PCM Programming Reference
- Conventional LUA Programming Reference
- Systems Management Programming Reference
- ACDI Programming Reference
- ECF SRPI Programming Reference
- X.25 Programming Reference
- Real-time Interface Co-Processor D.D. Prg. Ref.
- NTS/2 Publications
- APPC, APPN, & CPI-C Publications
- Redbooks (4)

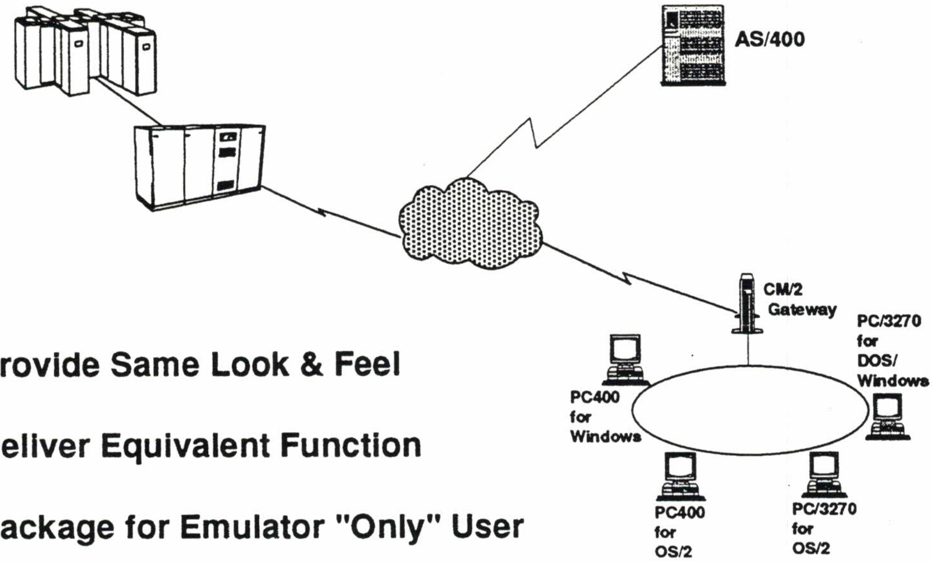
Communications Manager/2

► Today's Product - CM/2 1.11

➡ **Emulator Enhancement**

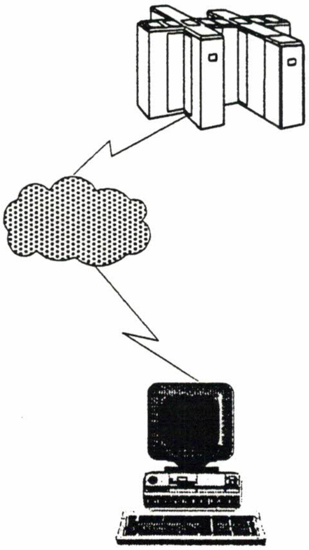
► Future Directions

Imperatives


AS/400
CM/2 Gateway
PC/3270 for DOS/ Windows
PC400 for Windows
PC400 for OS/2
PC/3270 for OS/2

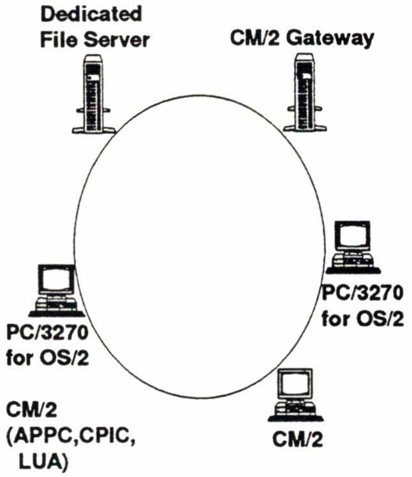
- Provide Same Look & Feel
- Deliver Equivalent Function
- Package for Emulator "Only" User

What's New?



- 32-Bit Implementation
- Additional Connectivities
  - IPX/SPX
  - TCP/IP
  - TN3270/TN5250
- User Productivity
  - Macros
  - Iconic Tool Bar
  - 3-D Hot Spots
- 32-Bit APIs
  - EHLLAPI
  - SRPI
- Dynamic Data Exchange(DDE)

Configuration Options

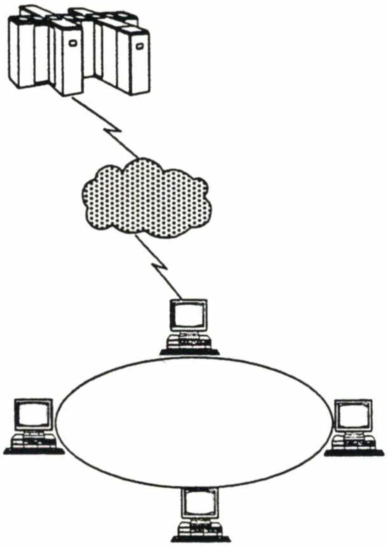

Dedicated File Server
CM/2 Gateway
PC/3270 for OS/2
PC/3270 for OS/2
CM/2 (APPC,CPIC, LUA)
CM/2

- If Gateway Required
  - Install CM/2
- If Additional Connectivities Required
  - APPC, CPIC, LUA
  - Install CM/2 and PC/3270 for OS/2

Communications Manager/2

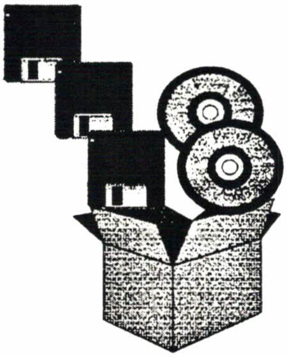
- Today's Product - CM/2 1.11
- Emulator Enhancements
- ➡ Future Directions

Future Directions - Function



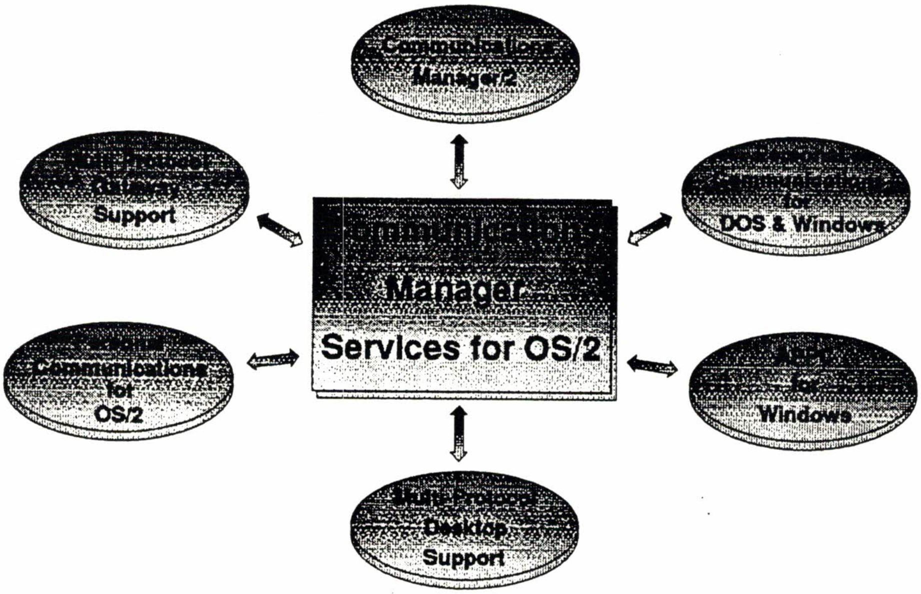
- SNA/APPN Enhancements
  - Transmission Priority
  - High Performance Routing
  - Dependent LU Requester
  - LU6.2 Enablement
    - Session Level Encryption
    - Sync.Point
- Connectivity Enhancements
  - SDLC
  - Multipoint
- 32-Bit Exploitation
  - Communications API's
  - System and System Services APIs
- Adapter Support
  - Deep Adapter Enablement
  - Shallow Adapter Enablement

Future Directions - Packaging



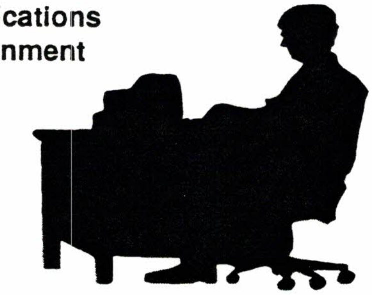
- Separate Emulators
- Gateways
- Programming Interfaces
- Connectivities

Future Directions


Communications Manager/2
Multi-Protocol Gateway Support
Communications for DOS & Windows
Communications Manager Services for OS/2
Communications for OS/2
for Windows
Multi-Protocol Desktop Support

32-Bit Application Development

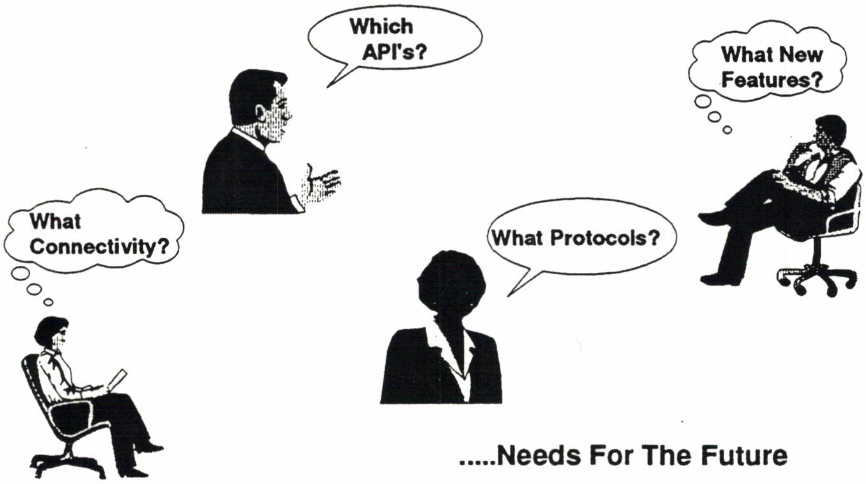
- Expressed Concerns
  - C Compiler Support
  - 16-Bit System Semaphores
  - Awareness of Underlying Environments
  - Migration of Applications to 32-Bit Environment



- Possible Priority for 32-Bit API's
  - EHLLAPI
  - CPI-C
  - LUA
  - APPC
  - X.25
  - SRPI
  - ACDI

OS/2 for Power PC - Communications

Major Application Requirements Today......


Which API's?
What New Features?
What Connectivity?
What Protocols?

.....Needs For The Future

Communications Manager/2

The Power of

The Communications Leader for OS/2
IBM
OS/2
Communications Manager/2
Version 1.1 CD-ROM

Personal Networking!

SYS 0107

Introduction to DCE Administration

No paper available. Please contact speaker

this page intentionally left blank

LAN 0204

Fine Tuning OS/2 LAN Server

Len Brissette

IBM

LAN 0204

OS/2 LAN SERVER PERFORMANCE TUNING

Len Brissette
IBM Personal Systems Products
Austin, Tx

Agenda

LAN SERVER PERFORMANCE

- PERFORMANCE ENHANCEMENTS
- CAPACITY ENHANCEMENTS
- TUNING CONSIDERATIONS
- PERFORMANCE COMPARISONS

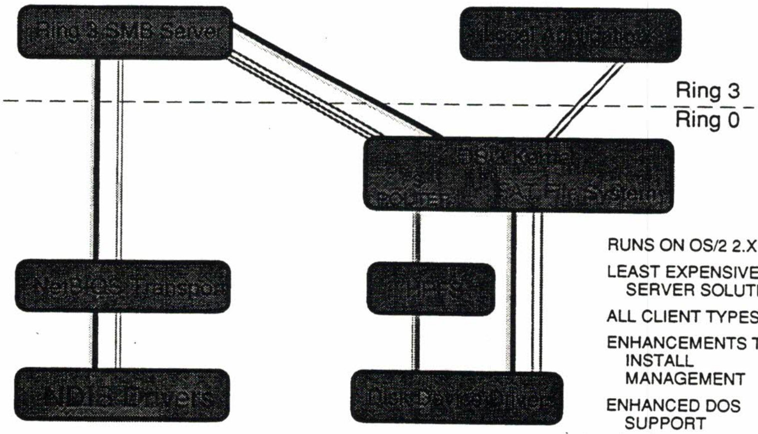
PSP LAN SYSTEMS

PERFORMANCE ENHANCEMENTS

- ENABLE FOR > 16 MB MEMORY
  - *LAN ADAPTER SUPPORT REQUIRED*
  - *LARGE CACHE, HEAP, OPEN FILE INFO*
- DOS CLIENT-SIDE CACHEING
  - *NEW DOS CLIENT CODE*
- SIDEBAND SUB-PROTOCOL
  - *IBM EXCLUSIVE*
  - *IMPROVES TRANSMIT BUFFER-SIZED/SMALLER I/O*
  - *RUNS ON OS/2, DLS CLIENTS AND ADVANCED SERVER*
- RING-0 TCP/IP

OS/2 LAN SERVER ENTRY

ENTRY SERVER DATA FLOWS


Ring 3 SMB Server
Local Applications
Ring 3
Ring 0
OS/2 Kernel
FAT File System
NetBIOS Transport
HPFS
NDIS Drivers
Disk Device Drivers

RUNS ON OS/2 2.X & 3

LEAST EXPENSIVE LAN SERVER SOLUTION

ALL CLIENT TYPES

ENHANCEMENTS TO INSTALL MANAGEMENT

ENHANCED DOS SUPPORT

Tuning Considerations

- Why Entry Server?
  - Dedicated Print Serving
  - Systems with Only FAT File System Partitions
  - Systems Sharing Only Serial Devices
  - Servers Only Verifying Logon

TUNING CONSIDERATIONS

- TOP 3 PERFORMANCE PARAMETERS for ANY OS/2 LAN SERVER
  - 1) CACHE
    - ► *Allocate as much cache as possible*
  - 2) NUMREQBUFS -
    - ► *3 Per Session*
  - 3) NUMBIGBUFS
    - ► *Check NET STATISTICS to see if server is running out*

ENTRY SERVER TUNING

- OBSERVE SERVER BEHAVIOR IN ACTUAL LOAD SITUATIONS
  - Resource exhaustion events are recorded
  - Use NET ERROR and NET STATISTICS commands
  - Using memory for cache will yield better performance than reserving request buffers or big buffers

PSP LAN SYSTEMS

OS/2 FILE SYSTEM TUNING

- ASSIGN CACHE FOR FAT PARTITIONS TO DISKCACHE.EXE
  - MAXIMUM CACHE SIZE: 14.4 MB
  - MINIMIZE IF NO SHARED FAT PARTITIONS: 64KB
- ASSIGN CACHE FOR HPFS PARTITIONS TO HPFS.IFS
  - MAXIMUM CACHE SIZE: 2MB
- ALLOCATE AS MUCH MEMORY FOR CACHE AS POSSIBLE
  - FOR HPFS 386 RECOMMEND MIN OF 6MB FOR 16MB MACHINE
- IF APPLICATIONS PERFORM BETTER WITH LARGE CACHES, ACCESS DATA FROM A FAT PARTITION WITH A LARGE CACHE ASSIGNED TO DISKCACHE

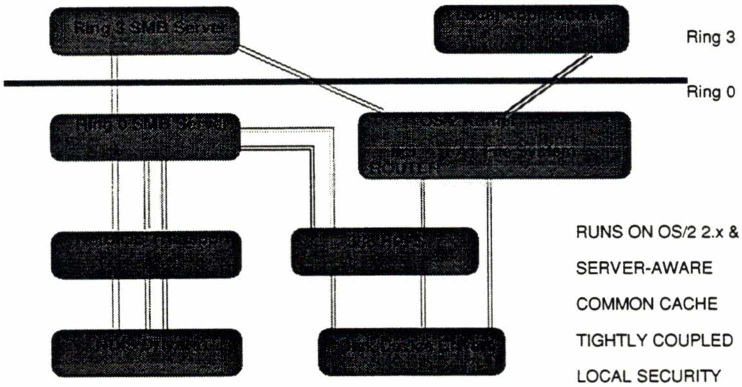
PSP LAN SYSTEMS

CACHE VS VDISK

- CACHE IMPROVES PERFORMANCE IF THE SAME DATA IS ACCESSED REPEATEDLY
- CACHE IS NOT AS HELPFUL IF DIFFERENT DATA IS REQUESTED AT EACH REQUEST
- A VDISK CAN BE SHARED JUST LIKE REAL DASD

OS/2 LAN SERVER ADVANCED

ADVANCED SERVER DATA FLOWS


Ring 3 SMB Server
Ring 3
Ring 0
Ring 0 SMB Server
ROUTER

RUNS ON OS/2 2.x & WARP

SERVER-AWARE

COMMON CACHE

TIGHTLY COUPLED

LOCAL SECURITY

RING 0

Entry vs Advanced Tuning Considerations

- The Advanced Server only supports file requests for files stored on an HPFS386 partitions; all other requests are still handled by the Entry Server
    - ► *FAT File System requests*
    - ► *Print*
    - ► *Serial Devices*
- The Entry Server must be tuned even on an Advanced Server for the resources supported by the Entry Server

PSP LAN SYSTEMS

ADVANCED SERVER FILE SYSTEM

- ASSIGN CACHE FOR HPFS386 PARTITIONS TO CACHE386.EXE

    - DEFAULT CACHE SIZE: 20% OR 60% OF AVAILABLE MEMORY
    - SUPPORTS CACHE SIZES GREATER THAN 16MB

- ADVANCED SERVER PERFORMANCE BENEFITS ARE ONLY REALIZED ON HPFS386 PARTITIONS AS MUCH MEMORY FOR CACHE AS POSSIBLE

    - RECOMMENDED MINIMUM OF 6MB FOR 16MB MACHINE

PSP LAN SYSTEMS

TUNING NTS/2 AND NETBIOS

- ALLOCATE NETBIOS RESOURCES (PROTOCOL.INI, NETBEUI_NIF SECTION)
  - SESSIONS
  - NCBs (2 TIMES SESSIONS up to 700 over all adapter cards)
  - NAMES
  - THESE PARAMETERS MUST BE GREATER THAN THE VALUES IN IBMLAN.INI ON THE APPROPRIATE NETx LINE
  - ALL NETBIOS APPLICATIONS MUST BE CONSIDERED (CM/2, DB/2, Lotus Notes, etc.)
  - NETBIOS "submit" applications get resources from IBMLAN.INI
- REDUCED NETBIOSRETRIES (on client) to IMPROVE LOGON PERFORMANCE
- INCREASE the T1 TIMER VALUE (on server) on WANs or SERVERS EXPERIENCING SESSION TIME OUTS

PSP LAN SYSTEMS

MEMORY MINIMIZATION ON CLIENTS

- OS/2 CLIENT
  - REDUCE NUMWORKBUF TO 5 IF NOT RUNNING MULTIPLE OS/2 SESSIONS
  - REDUCE SIZWORKBUF TO 2KB (MATCH THE SETTING AT THE SERVER)
  - REMARK OUT NETBIOS.OS2 IN CONFIG.SYS IF NO OTHER NETBIOS APPLICATIONS ARE RUNNING
  - SET PROTECTONLY TO YES (CONFIG.SYS) IF DOS SUPPORT IS NOT REQUIRED
  - SET MEMMAN TO SWAP,MOVE,SWAPDOS (CONFIG.SYS)
- DOS CLIENT
  - SET /BBC TO 0 (NO BIGBUFS)
  - LOAD CLIENT COMPONENTS IN HIGH MEMORY ON 386 (OR HIGHER) WORKSTATIONS (/HIM, /UMB, OR /EMS)
  - USE STANDARD DOS NETBIOS INSTEAD OF DOSBEUI/NDIS

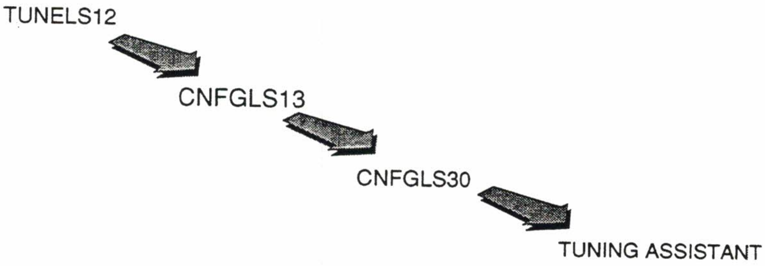
PSP LAN SYSTEMS

OTHER TECHNIQUES

- KEY DOCUMENTATION
  - LS NETWORK ADMINISTRATOR'S REF. VOL. 2
  - LS 4.0 PERFORMANCE WHITEPAPER
- OBSERVE THE SYSTEM
  - PERFORMANCE MONITOR/2 or Netview Monitor
  - SERVER ERROR LOGS AND STATISTICS
  - CACHE386 MONITORING UTILITY
- USE NETWORK ANALYZERS TO:
  - DETECT UNUSUAL APPLICATION BEHAVIOR
  - ISOLATE SOURCES OF NETWORK LOAD
  - CAPTURE ERRORS
  - MEASURE LINE UTILIZATION

PSP LAN SYSTEMS

TUNING TOOLS



TUNELS12

CNFGLS13

CNFGLS30

TUNING ASSISTANT

- LAN SERVER 4.0 INCLUDES TUNING ASSISTANT APPLET
- CAN USE CNFGLS30 FOR LS 2.0 OR LS 3.0
- IT WAS WRITTEN IN CONJUNCTION WITH THE DEVELOPMENT TEAM
- AVAILABLE ON OS/2 BBS or COMPUSERVE (within IBM on MKTTOOLS)

PSP LAN SYSTEMS

ENTRY SERVER CONSIDERATIONS

- FAT CACHE (DISKCACHE)
  - *14.4 MB Maximum*
  - *Threshold up to 64KB*
  - *OS/2 2.0 adds LAZY WRITE*
- HPFS CACHE (CACHE)
  - *2.0MB Maximum*
  - *Threshold up to 64 KB*
  - *LAZY WRITE*
- IBMLAN.INI
  - *NUMREQBUFS (3 per active client)*
  - *NUMBIGBUFS (2 per active client)*
- PROTOCOL.INI
  - *SESSIONS*
  - *NCBS*

ADVANCED SERVER CONSIDERATIONS

- HPFS386 CACHE SIZE
  - *System defaults to 20% or 60%*
  - *CACHE SIZE limited only be available memory in the system*
- CACHE386 MONITORING UTILITY
  - *Fine tune CACHE SIZE*
  - *Monitor cache hit ratio*
- NUMREQBUFS
  - *3 per session*
  - *Check NET ERRORS*

IF ALL ELSE FAILS...

- IBM PERSONAL SYSTEMS SERVICE AND SUPPORT IN DALLAS PROVIDES A FEE SERVICE FOR TUNING LAN SERVER INSTALLATIONS

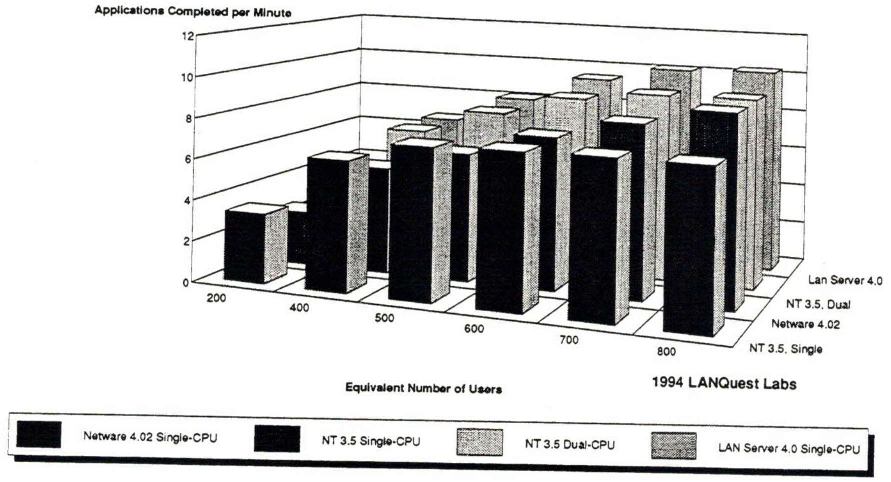
PERFORMANCE COMPARISONS

LANQUEST LABS:

"... IBM LAN SERVER 4.0 Advanced outperforms both Microsoft Windows NT Server v3.5 and Novell Netware v4.02 ... on both single and dual-CPU systems. ... LAN Server Advanced was 38% faster than Windows NT Server and 11% faster than Netware.:"

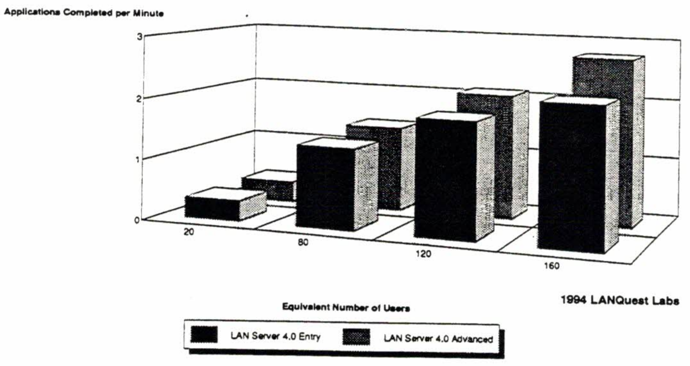
- PERFORMANCE BENCHMARK COMPARISON
  - NOVEMBER 1994
- COPIES OF THE REPORT CAN BE OBTAINED THROUGH 1-800-IBM-4FAX. DOCUMENT 2014 and 2347.

LAN SERVER PERFORMS!



PSP LAN SYSTEMS

LS ADVANCED vs LS ENTRY



PSP LAN SYSTEMS

NSTL COMPARISON

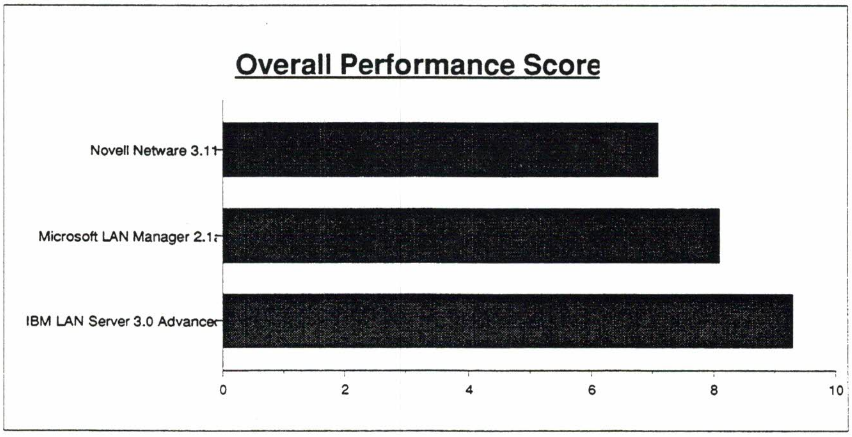
- National Software Testing Laboratories

"LAN Server had the highest Overall Performance Score on NSTL's test configuration across all five (WordPerfect 5.1, Lotus 1-2-3, dBase IV, XCOPY, cc:Mail 3.2) application - based tests. (Comparing LAN Server 3.0 Advanced, LAN Manger 2.1a, and Netware 3.11)"

- **Network Operating System Performance Comparison**
  - **November 1992**

PSP LAN SYSTEMS

PERFORMANCE COMPARISONS



PSP LAN SYSTEMS

SUMMARY

- PERFORMANCE ENHANCEMENTS
- CAPACITY ENHANCEMENTS
- TUNING CONSIDERATIONS
- PERFORMANCE COMPARISONS

PSP LAN SYSTEMS

LAN 0206

OS/2 Distributed Computing Environment Overview

Sanford Rockowitz
Minaret Software
Daly City, Calif.

Copyright © 1995 by Minaret Software. All rights reserved.

ABSTRACT

The Open Software Foundation's Distributed Computing Environment (DCE) provides an industrial-strength infrastructure for creating and using applications in heterogeneous environments. Its fundamental goal is to mask the complexity of the network from both the application programmer and the end user. Originally developed for UNIX, implementations now exist for operating systems as diverse as OS/2 and MVS. IBM has adopted DCE as a key technology for integrating its many operating systems and platforms. Key components of DCE are Remote Procedure Call, Directory Services, Security Services, Distributed Time Services, Threads, and the Distributed File System.

This paper provides an overview of DCE, outlines how DCE fits into the range of middleware options, and reviews the implementation of DCE for OS/2. It also suggests some guidelines for deciding when DCE is an appropriate solution for application distribution.

Why DCE?

Once upon a time, all computing was "local" in the sense that it happened on a single machine. At first, these machines were mainframes, then mini-computers, and eventually PCs. In these purely local environments, there were many simplifying assumptions we rarely even realized we were making. Among them:

- There was a single data representation. Integers were either big-endian or little-endian, character data either (our local version of) ASCII or (our local version of) EBCDIC.
- We could log on once, and having logged on had an identity that all programs could rely on.
- Components of an application communicated using procedure calls.
- All files were in a single file system, accessed in uniform way.

As computing becomes distributed, these and other simplifying assumptions no longer hold. Consequently, application programmers expend enormous time and effort creating their own distributed computing infrastructure. By some estimates, as much of 80% of the code in a distributed application can be "plumbing" not specific to the application at hand. This plumbing is required to address issues such as the following:

- Distributed computing environments commonly contain a variety of computer types and multiple operating systems. Even the data representation can vary from system to system.
- Client and server components must communicate over the network.
- Client and server components can fail independently, or the communications substrate can fail. Applications need to be able to recover from such situations.
- Client and server components do not have access to the same shared memory, so data cannot be passed by simply passing a pointer. Generally, a special transfer format is required to exchange data over the network.
- Components have to find each other. Applications must find their servers, and more generally users have to be able to locate resources.
- Components of a distributed computing environment must be able to trust each other and trust that their communications have not been compromised.
- Bottlenecks can limit scalability. Solutions that work for tens of systems on a LAN may not scale to hundreds or thousands of systems on a WAN.

Distributed environments also create complexity for end users, who must learn how to navigate the network and new ways of doing familiar tasks. For example, to copy a file under OS/2 we use the COPY or XCOPY command. If that file is on a LAN Server machine, we must first issue a NET USE command before copying. If it's on a NFS system, we have to MOUNT the file system as a drive. And if its being accessed via FTP, there's the entirely different syntax of OPEN and GET.

What is DCE?

The Distributed Computing Environment (DCE) from the Open Software Foundation is an integrated set of facilities for creating and using distributed applications in a heterogeneous environment. The key components of DCE are:

- **Remote Procedure Call**, which enables components to communicate using the well understood method of procedure calls.
- **Directory Services**, which enable components to find one another.
- **Security Services**, which enable components to trust one another, and trust that their communications have not been compromised.
- **Distributed Time Services**, which enables components to agree on what time it is.
- **Threads**, which provides an operating system independent set of facilities for controlling parallel execution of tasks.
- The **Distributed File System**, which enables users to access files as if they were all in one large logical file system.

The fundamental purpose of DCE is to enable us to operate in the network computing environment with much the same simplicity as we once operated in the local computing environment. The network should be perceived as a single system, not a collection of independent parts. In DCE terminology, this is referred to as *transparency*, which takes several forms.

- Application transparency means that local and remote services and files are accessed in an identical manner. Programmers use a similar procedure call interface to invoke both local and remote services. Users can access local and remote files using the same commands.
- Location transparency means that users can access resources by referring to their logical names, without knowing their physical location. Given location transparency, resources can be moved as required without affecting programs and people who use those resources. Another aspect of location transparency is that users have the same DCE identity no matter what machine they are using, and that resources have the same name no matter what machine they are accessed from.
- Replication transparency means that copies of files and other data are automatically maintained for performance, invisibly to the programmer and user.

Other key objectives of DCE are high performance, reliability, and scalability. Through the use of replicated servers, transparent caching, and multithreaded operation, the design and implementation of DCE is oriented toward support of computing environments with thousands of machines. The machine independent client/server interface provided by remote procedure call, along with the ability to move servers from machine to machine without changing their logical name, makes it possible to move a server to more a powerful machines transparently to the clients that use it.

WHO'S IMPLEMENTING DCE?

DCE may be great technology, but unless it has broad industry endorsement a system of DCE's complexity does not warrant our attention. While support is not universal, DCE is being widely adopted. DCE began as a UNIX facility, but now exists on systems ranging in scale from PCs running Windows to mainframes running MVS. IBM has adopted DCE as a strategic solution to interoperability across its diverse platforms. DCE implementations exist for all significant versions of UNIX, including AIX from IBM, UnixWare from Novell, SCO UNIX, HP/UX, OSF/1, Siemens SINIX, DEC Ultrix, AT&T GIS, Fujitsu UXP/DS, Data General AViion, and Sun Solaris. There are implementations for DEC VMS, Tandem NonStop, Bull GCOS, and Siemens Nixdorf BS2000. IBM has implemented or is implementing DCE on MVS, VM, AS/400, and OS/2, as well as AIX, and is working on merging DCE services into LAN Server. Major database vendors including Informix, Oracle, Sybase, Ingres, and IBM are integrating DCE facilities into their products. The Encina TP monitor for AIX and

HP/UX is based on DCE, and DCE facilities are being incorporated into CICS.

The relationship of Microsoft and Windows to DCE is somewhat complex. Microsoft has implemented DCE remote procedure call in Windows NT, Windows 95, and even Windows 3.1. It is a core technology that underlies OLE. However, apparently to avoid licensing fees, Microsoft chose to create its own implementation of remote procedure call based on the DCE specification instead of obtaining the reference code from OSF. Microsoft implemented only the RPC portion of DCE; it uses its own services instead for directory and security, though it is possible for a Windows DCE application to use a DCE cell directory server running on a non-Windows machine. Thus, while Microsoft claims "compatibility on the wire" with other DCE implementations, it is generally not possible to simply take a DCE application and recompile it for a Windows platform. However, true DCE implementations do exist for Windows 3.1 from Gradient and for Windows NT from DEC.

The relationship of Novell to DCE is also complex. While DCE is directly supported in UnixWare, it appears that Novell sees it as a competitor to Netware technology, and will implement only enough of DCE in Netware to allow linking to external hosts. Novell offers a DCE gateway to its Tuxedo TP monitor.

The final problematic company is Sun. As the UNIX wars have abated, there has been a rapprochement between Sun and OSF. As a result, Sun now offers full support for DCE on Solaris, along with its own ONC.

SOME HISTORY

Technologies like DCE are created by organizations with interests and agendas, so it's helpful for understanding a product to understand who produced it and how.

The Open Software Foundation was created in 1988 by a collection of UNIX vendors who had as their objective the promotion of "open, multi-vendor computing". Translated, OSF was intended as a counterweight to the then dominance of UNIX by Sun and AT&T. Its primary members were IBM, HP/Apollo, DEC, Groupe Bull, Siemens/Nixdorf, and Hitachi. OSF created several products, among them the Motif graphical user interface (1989), the OSF/1 operating system (1990), the Distributed Computing Environment (1991), and the ill-fated Distributed Management Environment (1994).

At the time DCE was created, OSF used an interesting approach to creating software. Instead of starting from scratch, it chose to adapt and integrate existing software products, what it called the "best of breed" approach. Having identified a product area, OSF solicited requests for technology for the various components. Components were chosen, then integrated on a reference platform. The integrated source code was given to vendors to adapt and enhance for their specific platforms. In the case of DCE, the remote procedure call technology was based on Network Computing System (NCS 2.0) from HP's Apollo division. Cell Directory Services came from DEC's dns, global X.500 directory services from Siemens DIR-X. Security authentication is based on Kerberos from MIT's Project Athena, while authorization uses the access control list mechanism from HP/Apollo's PasswordEtc. The Distributed Time Services are based on DEC's dts, and the Distributed File System comes from Transarc and is based on the Andrew File System developed at Carnegie-Melon. IBM integrated the initial DCE release on AIX.

The "best of breed" approach has both advantages and disadvantages. It is based on existing, real solutions that are known to work. OSF thereby avoided the problem so common to the standards process of designing "ultimate" systems so feature heavy that they're never built. Because vendor implementations are based on common reference code, they generally avoid interoperability problems resulting from differing interpretations of a specification.

The "best of breed" approach also has disadvantages. DCE sometimes has a cobbled together feeling. Facilities overlap, and there are inconsistent APIs and user interfaces. Also, limiting assumptions of the initial products tend to creep through. For example, while in principle DCE is language and transport neutral, in fact it reflects its UNIX origins by a clear bias towards C and TCP/IP.

MIDDLEWARE ALTERNATIVES

To address the problems of distributed computing, a new class of software has emerged, or perhaps it's more accurate to say that it has recently been given a name, that of *middleware*. Middleware aims to shield the application programmer from the underlying details of distributed computing. DCE is a premier example of this class of software. There are alternative middleware paradigms, and even competing implementations of the remote procedure call paradigm, so a brief overview of the middleware universe may help situate DCE.

The *conversational* middleware model is epitomized by protocols such as CPI-C or TCP/IP sockets. It is fundamentally a low level, synchronous protocol: a packet of information is sent and the sender waits for a packet to arrive. Both parties of a transaction must be active for communication to occur. Interpretation of packet contents is entirely the responsibility of the application, as is the process by which packets are organized into a larger unit of work.

The *message oriented middleware* (MOM) model is similar to the conversational in that the message is the basic unit of implementation, but here the messages are at a more abstract level. Messages are well defined entities that are managed by the middleware layer. Whereas the conversational model is fundamentally synchronous, MOM is asynchronous. The recipient of a message need not be active when a message is sent. Among MOM's key attributes are persistent message queues, store and forward routing, guaranteed asynchronous delivery, and priority messages. MOM is well oriented to loosely coupled systems and to implementing office workflow systems. Key products in this area are PIPES from Peerlogic and MQSeries from IBM.

Remote procedure call based middleware products can be thought of as falling into three groups, which I'll label primitive, classical, and postmodern. DBMS stored procedures are a primitive form of remote procedure call. So long as your application is entirely implemented within a single vendor's universe, stored procedures are a powerful tool for distributing function. Their drawback is that they are by definition useful only if what you need to do stays within the boundaries of the DBMS.

Classical remote procedure call attempts to replicate the full functionality of procedure calls as found in general purpose languages such as C. The two dominant protocol suites are DCE from OSF and ONC from Sun. Although this paper is devoted to DCE, a few words on Sun RPC may be in order. ONC is an extension of the Internet Remote Procedure Call protocol (which Sun contributed). ONC provides many of the same types of services as DCE, including remote procedure call, directory services (the Yellow Pages) and security. (While the services are similar conceptually, the two protocol stacks cannot interoperate.) Sun RPC is the protocol on which many Internet services, including NFS, are based. While it can be argued that DCE is in many ways a more robust and sophisticated protocol, that both these protocol stacks exist is fundamentally a reflection of the UNIX wars of the late 1980's, with Sun and AT&T on the one hand supporting ONC, and nearly all the other UNIX vendors championing DCE. If you decide that what you need is classical RPC, your choice between ONC and DCE will probably reflect the environment you're in. If it's fundamentally a Sun network, you'll likely choose ONC, and if its fundamentally a non-Sun environment you'll want to opt for DCE. (It should be noted that using Solaris does not prevent you from using DCE. A full DCE implementation is available from Transarc and marketed by Sun.)

The third major RPC protocol suite is the OSI ROSE (Remote Operation Service Element), with its underlying data definition language ASN.1. While some application domains make heavy use of ASN.1, OSI has is basically an also ran in the protocol marketplace.

If DCE and ONC represent classical RPC, distributed objects can be thought of as RPC in its postmodern form. Instead of executing functions remotely, we're invoking methods. There's considerable overlap in functionality between remote procedure call and distributed objects, and in a later section we'll take a closer look at the relationship between DCE and distributed object standards.

At some level, of course, DCE is built on top of messaging. However, RPC hides the definition of data packets, and organizes the packets into procedure calls. Conversely, MOM can be built on top of RPC. The question is which model the application programmer sees.

Cells

Having set the stage, let's now look at the DCE technology itself. Though the printed page constrains us to describe the various DCE components in some particular order, bear in mind that the components are highly interrelated. For example, Directory, Security, and Distributed Time services are all implemented using remote procedure calls, and remote procedure calls in turn use Directory, Security, and Time services.

A key DCE notion is the *cell*. A cell is a collection of computers that closely interoperate. It is the basic unit of DCE administration. Logically, machines in a cell share a single directory, have a single arbiter of security, and share a common time. Usually, though not necessarily, machines in a cell are in the same geographic area. Cells can range in size from 1 machine up to thousands. They can include multiple LANs linked by WANs. DCE is designed to optimize intra-cell communication. By default, DCE entities can find and communicate only with other DCE entities in the same cell.

Cells in turn can be linked through use of global directory services, either X.500 or the Internet Domain Name System. Cells have names, which are either X.500 names or DNS names. Figure 1 illustrates a cell. Note that a machine in a cell can run both client and server components of multiple applications. In DCE, "client" and "server" designate program roles, not specific machines.

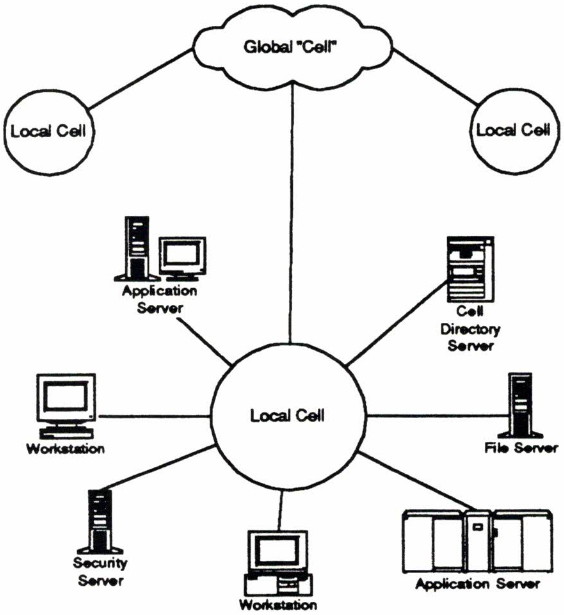


Figure 1

REMOTE PROCEDURE CALL

Procedure calls are a well understood paradigm for managing program complexity. Remote procedure calls extend this familiar and robust programming paradigm to the network environment.

In the usual non-distributed programming environment, calling procedures are directly linked with the procedures they call, or invoke routines in a dynamic link library or transient library. With remote procedure call, the calling and called procedures can reside on different machines. This introduces several complications:

- The caller cannot branch to the callee, since the callee is in a different address space.
- Pointers cannot simply be passed between callers and callees. The data that pointers point to must also be copied.
- The machines may have different data representations. The classic example is of integers. Some architectures, such as the Intel x86, store integers in little-endian form, whereas others, such as S/370 mainframes, store it in big-endian form. Data must be converted. A more complicated example of data conversion is character data, since character strings can change size in translation from one encoding to another.
- Procedure calls can fail due to network failures or operating system failures at the callee system, which are outside the control of the application.

The Remote Procedure Call facility of DCE manages these complexities in a manner that largely masks them from the application. Among the key aspects of DCE RPC:

- The remote procedure calls provided by a server are specified using a C-like Interface Definition Language (IDL).
- Procedure calls are encoded for transmission in a serialized format known as Network Data Representation (NDR). This is transparent to the programmer, as conversion from the application's data structures to the serialized form occurs automatically. A significant aspect of the encoding method is its "receiver makes right" approach, which avoids unnecessary conversions.
- Remote procedure call is integrated with Directory Services to enable clients to find servers.
- Remote procedure call is integrated with Security Services to guarantee authenticity, integrity, and privacy of communications
- Remote procedure call is nominally transport independent, but in practice nearly all implementations require TCP/IP.

Figure 2 contains a sample interface definition for a server named demo. demo provides one function, silly, whose purpose is to illustrate as much about RPC in as little space as possible.

silly has 3 parameters:

- cli_greeting - a string that the client sends to the server
- srv_reply - a string that the server sends back to the client
- pd - pointer to structure DATA, which is transmitted to the server and returned to the client. DATA contains:
  - a vector of 4 integers
  - a multiplier; The server multiplies each integer in the vector by the multiplier. For illustrative purposes, the multiplier is passed as a string. It is the server's responsibility to convert it to an integer.

```
uuid(004c4b40-02fb-1d7d-b062-02608ca974d0),
version(1.0)
]

interface demo
{
  const long STR_SZ = 128;
  const long ARRAY_SZ = 4;

  typedef struct {
               long   a[ARRAY_SZ];
      [string] char   multiplier[STR_SZ];
  } DATA;

  void silly
  (
    [in]         handle_t h,
    [in,out]     DATA *   pd,
    [in,string]  char cli_greeting[STR_SZ],
    [out,string] char srv_reply[STR_SZ]
  );
}
```

Figure 2

Several aspects of the interface definition warrant note:

- The interface definition has a header and a body. The header is contained in square brackets at the start of the definition.
- The header begins with a machine generated universal unique identifier (UUID), which uniquely identifies the interface. The algorithm used to generate the UUID is designed to ensure that the UUID is globally unique. The UUID is used internally by DCE when searching for a server.
- The interface definition provides for version control. Every interface has a version number, which consists of a major number and a minor number, separated by a decimal point. Changes to the major number are used by application developers to indicate interfaces that have changed so much as to be incompatible. Changes to the minor number are used to indicate upwardly compatible revisions, such as adding an additional procedure. A client program having a given major and minor version number will be compatible with a server having the same major number and a minor number at least as great as the client's.
- Though our example defines only one remote procedure (silly), an interface definition body can and normally does contain definitions of many functions.
- Parameter *attributes* in square brackets provide information needed to implement the procedure call remotely. *in* and *out* indicate whether parameters need to be copied from the client to the server or returned from the server to the client. *string* indicates that a parameter is a null delimited character string, only bytes up until the terminating null character need to be copied.
- The remote procedure call includes a *binding handle* of type handle_t. This is a handle to information used by the DCE runtime to identify the network address and endpoint of the client and server and the protocol sequence they are using. (Note: In some circumstances, this binding information can be managed implicitly and need not appear in the interface definition.)

As figure 3 illustrates, the interface definition is processed by an IDL compiler to produce several components. These include stub routines that are linked with the client and server, and definition files (typically C header files) that define the interface between the client and server application code on the one hand and the generated procedure call stubs on the other.

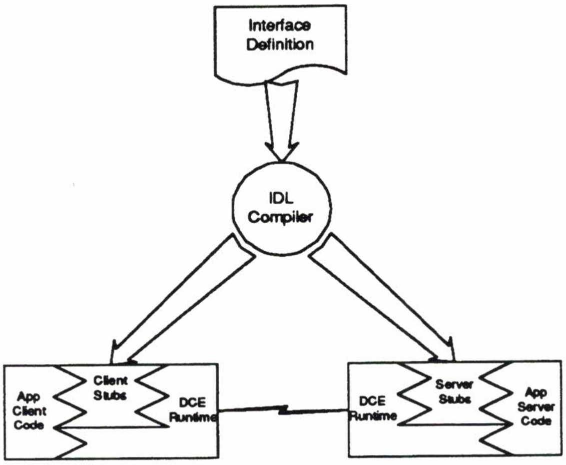


Figure 3

Figure 4 contains a fragment from the generated C header file for the demo server. The header file defines the interface between the client and server application routines and their respective stubs:

```
          #define STR_SZ (128)
          #define ARRAY_SZ (4)

            typedef struct  {
             idl_long_int a[4];
       idl_char     multiplier[128];
                 } DATA;

       extern void  DCEAPI silly(
        /* [in] */      handle_t h,
        /* [in,out] */  DATA *   pd,
/* [in] */      idl_char cli_greeting[128],
  /* [out] */     idl_char srv_reply[128]
                    );
```

Figure 4

To note:

- Types idl_long_int and idl_char map through typedefs in the standard DCE header files to language constructs appropriate for the particular compiler and platform, such as long and char.
- DCE consts are converted to C #defines.

In addition to the interface definition (IDL) file, the IDL compiler can optionally take as input an Attribute Configuration File (ACF). Unlike the IDL file, which specifies the interface between client and server, the ACF specifies properties that are local to the client or server, such as whether marshaling is to be done inline or by using subroutines. Whereas an interface has a single IDL file, each client and server implementing the interface can have a different ACL file.

Figure 5 shows what happens when a remote procedure call executes.

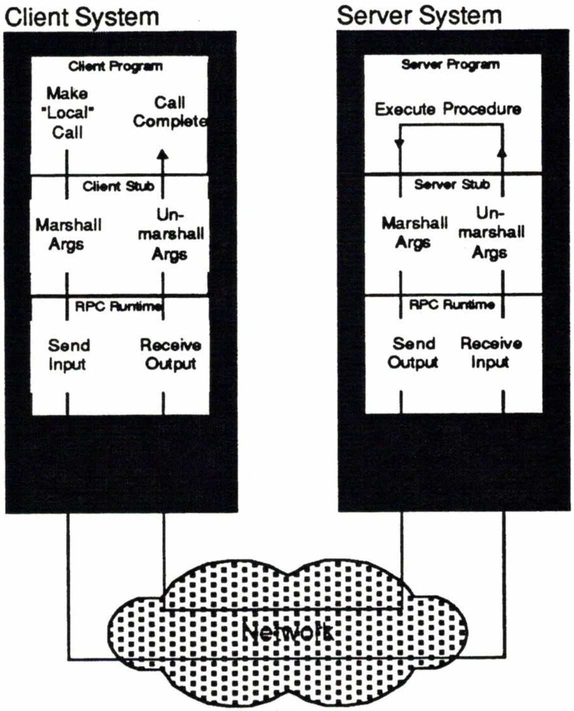

Client System
Client Program
Make "Local" Call
Call Complete
Client Stub
Marshall Args
Un-marshall Args
RPC Runtime
Send Input
Receive Output
Server System
Server Program
Execute Procedure
Server Stub
Marshall Args
Un-marshall Args
RPC Runtime
Send Output
Receive Input
Network

Figure 5

The steps are as follows:

1) The client makes a local procedure call to the stub procedure that is the surrogate for the remote procedure.
2) The stub marshals the arguments. Marshaling converts parameters into byte strings using Network Data Representation (NDR) format. If necessary, pointers are dereferenced. That is, the data referenced by a pointer is itself copied into the NDR data stream.
3) The stub calls the RPC runtime, which in OS/2 is in a DLL.
4) The client's RPC runtime transmits the procedure call over the network to the RPC runtime in the server.
5) The RPC runtime in the server dispatches a call to the appropriate server stub.
6) The server stub unmarshals the arguments, constructing data structures that mirror those of the caller in the client program. Any pointers now reference areas in the server's memory.
7) The server stub calls the server function that will do the work of executing the remote procedure. From the point of view of the application server function, it has been invoked by a normal, local call.
8) The server procedure executes and returns to its caller in the stub.
9) The server stub marshals results and passes them to the RPC runtime on the server's system.
10) The server's RPC runtime transmits results over the network to the client RPC runtime, which returns them to the client stub.
11) The client stub unmarshals arguments. It updates any client data structures to reflect changes made by the server, and performs procedure call return to the client program.

DIRECTORY SERVICES

In a distributed environment, every entity requires a distinct, unambiguous name. Entities include people, servers, physical devices such as printers, and logical devices such as print queues. The logical database used for storing the names of entities and their attributes (e.g. where a server is physically located) is called the *namespace.*

Cell Directory Services (CDS) provides a unified namespace for all objects in the cell. Security and distributed file system names are logically incorporated into the namespace through the construct of *junctions*. (As illustrated in figure 6, the security namespace is junctioned off directory sec in the cell namespace, and the distributed file system is junctioned off directory fs.)

Cell Directory Services is integrated into Remote Procedure Call. Servers can register their names in the namespace along with their location, and clients can look up server names in the namespace to obtain the location (i.e. machine) at which they can be found. While logically there is a single directory for an entire cell, for performance CDS uses a replicable, partitionable database.

It's hard to talk about things without naming them, so let's take a moment to see how objects in the cell namespace are named. The namespace is hierarchically organized into directories using the familiar pathname conventions. The following prefixes have special meaning:

/.../	Indicates the global root. Names beginning with /.../ are universal. Users anywhere in the world use the same global name.
/.:/	Indicates the cell root. Names beginning with /.:/ are fully qualified within the cell.
/:/	Indicates a file system name. This is logically equivalent to /.:/fs/

Figure 6 shows some examples of DCE names.

One important distinction to keep in mind is to avoid confusing an object with its name. File names, for example, may appear in the namespace, but the file itself resides in a file system.

FINDING A SERVER

Not only must objects have names, given a name we must be able to locate the object. Associated with each name in the directory is a set of attributes. For server names, this includes the interface provided by the server and the system on which it is found.

Suppose an application I'm running wants to access server /.../minsoft.com/servers/demo. Figure 7 shows what happens.

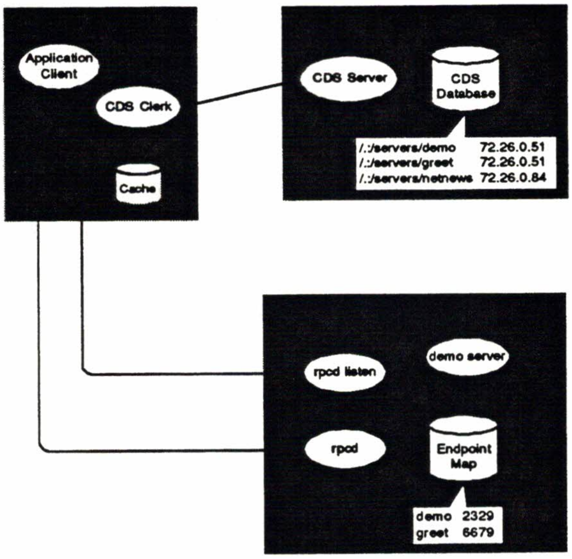


Figure 7

The steps are as follows:

1) The application makes a call to the RPC runtime, which in turn makes a request of the Cell Directory Services Clerk on my local machine. The Clerk is running as a daemon process.
a) For performance, the CDS Clerk first consults its local cache.
b) If it doesn't already know the answer, the CDS Clerk accesses the CDS Server that's in the cell.

c) For names outside the current cell (and as yet uncached), the CDS server uses a Global Directory Agent (not shown) to resolve names in other cells. (Global Directory Services are not yet available in the OS/2 Cell Directory Server. However, OS/2 clients can use non-OS/2 Directory Servers that do have Global Directory Agents.)

The CDS Clerk returns to the application the host on which the server resides.
2) The application (or more precisely, the RPC runtime on the client system) contacts the rpcd daemon on the server's machine (again using RPC). rpcd consults its local endpoint map to determine the endpoint at which the server is located, and returns that endpoint to the application.
3) The application contacts the server directly, by addressing its messages to the specified endpoint at the specified host.

But how does the namespace know where a server is located? Typically, the server tells it! As part of server initialization, the server creates or updates an entry such as */.:/servers/demo* in the CDS namespace. Among the attributes stored for the entry are the host address of the machine on which the server is located, the interface it exports (specified using the interface UUID described above), and the communication protocols that the server can use (e.g. DCE connection oriented protocol over TCP/IP). Collectively, this information, along with the endpoint on the local machine, is referred to as *binding* information, and the process by which a server publishes this information in the namespace is referred to as *exporting the bindings* or *advertising the bindings*.

Strictly speaking, the endpoint to be used by an interface can be included in the interface definition header and exported into the namespace. This technique of staticly assigned well known endpoints is in fact used for certain core servers. In particular, rpcd is assigned a well known endpoint. In general it is preferable to assign endpoints dynamically at the time of server initialization and register them in the local endpoint map. Otherwise, the implementer of every server would have to be aware of the dedicated endpoints used by every other server.

It is also possible to dispense with the cell directory server and hard code the identity of the server machine in the client application or pass it to the client at startup. This can be useful in simple environments where resources are an issue and to simplify development, but by doing so you forgo much of the location transparency afforded by DCE.

SECURITY

Security in distributed systems is commonly both onerous and inadequate. It's onerous because each subsystem has its own security mechanism requiring a separate logon. It's frequently inadequate for two reasons. First it's implemented by programmers whose expertise and concern lie in the system they are developing, not security. Passwords are often transmitted in the clear over networks. Though clients authenticate themselves to servers, it's infrequent that servers authenticate themselves to the clients that use them. Second, the very onerousness of the security mechanisms leads users to behaviors that compromise system integrity, such as simple, infrequently changed passwords, identical passwords used for multiple systems, and automation of access to "secure" systems using hardcoded passwords laced throughout logon scripts on unsecure machines.

With DCE, users log on once to the DCE environment and acquire a single, verified identity upon which all DCE applications can rely. (This DCE identity is generally distinct from the user's login identity on the local timesharing system.) Instead of unencrypted passwords, encrypted *credentials* (called *tickets*) are transmitted over the network. To further frustrate cryptographic attacks, these tickets are time limited.

DCE security services fall into three groups:

- *Authentication* services ensure that entities (users, servers, etc.) are who they claim to be.
- *Authorization* services regulate access to resources based on an authenticated entity's verified identity.
- *Secure communication* services allow entities to communicate secure in the knowledge that their communications have not been tampered with. This takes two forms. *Integrity verification* ensures that data has not been altered in transit, i.e. that what is received is what was sent. *Confidentiality* goes even further; not only can eavesdroppers not tamper with what is being sent, they can't even decipher it.

In DCE terminology, security is implemented using a set of servers and facilities. Authentication is based on Kerberos Version 5, from MIT's Project Athena. The security server program (secd) is logically composed of multiple servers: the Registry Server, the Authentication Server, the Ticket Granting Server, and the Privilege Server. The Registry Server manages the security database, which keeps a record of principals and passwords. A principal is any entity on a computer network capable of proving its identity, such as a user, a client program, or a server. The Authentication Server authenticates principals using secret key encryption and issues encrypted credentials called tickets. The Authentication Server accesses the Registry Server as necessary. When an authenticated user wishes to access a particular server, it passes its ticket to the Ticket Granting Server, which returns to it a ticket for use in accessing that particular server. Associated with these servers is an extension to Kerberos, the Privilege Server, which records group memberships and provides for intercell authentication.

The login facility, which runs on a user's workstation, allows the user to log in once to DCE. On OS/2 the user runs the *dcelogin* command to invoke the login facility. On most other DCE systems, not constrained by the 8.3 naming convention, the command is *dce_login*.

Applications can choose to restrict access to the resources they manage. The Access Control List facility can be used by applications to determine a principal's ability to access a resource. Authorization can be granted to individuals, users, or groups. The ACL facility conforms to the Posix 1003.6 Access Control List standard.

Auditing capabilities were not originally part of DCE Security Services. However, the most recent DCE release (1.1) allows administrators to track events in the security and time subsystems. Events include use of authentication and authorization, update of the security database, security related actions taken by system administrators, and changes to the cell configuration.

Another feature new in DCE 1.1 is the Generic Security Service API (GSS-API), which enables non-DCE applications to use DCE security services. The GSS-API is as defined in Internet RFC 1508 and 1509, with extensions.

HOW KERBEROS WORKS

The Kerberos authentication mechanism is one of DCE's most elegant aspects. This section gives a brief, simplified overview of how secret key encryption works. Readers uninterested in such details can safely skip the explanation. What's important to know is that Kerberos provides a nearly bulletproof security mechanism that allows entities in a DCE cell to trust one another even though the network itself is insecure. Security is symmetric. Clients prove who they are to servers, and servers prove who they are to clients. They do so by transmitting messages that cannot be compromised by eavesdroppers.

For simplicity, we ignore for the moment the extensions to Kerberos for group memberships provided by the Privilege Server.

The following key Kerberos concepts are helpful to keep in mind for understanding how Kerberos works:

- The basic Kerberos server consists of 3 servers that reside on the same, physically secured machine: the Registry Server, the Authentication Server, and the Ticket Granting Server.
- The authentication system verifies the client to the server (and vice versa). There is no need for application servers to know the client's password.
- The Authentication Server validates an entity's identity and gives it a credential (a.k.a. ticket) granting access to the Ticket Granting Server.
- The Ticket Granting Server gives the entity tickets allowing access to individual application servers. The entity in turn gives the ticket to the application server.
- Tickets issued by the Authentication Server and Ticket Granting Server are encrypted using the key that the target server has shared with the authentication system. Neither the client nor any eavesdropper on the network can tamper with the tickets.
- Passwords are not normally passed between client and server, but instead are used as encryption keys during logon. This protects both sides of the exchange. The only time that passwords are actually passed is when the password is being updated, in which case it transmitted within an encrypted packet.
- To further frustrate attacks on the system, all exchanges after the initial Authentication Server logon access use time limited transaction keys generated on the fly.

Figure 8 shows how Kerberos authentication works. It illustrates the sequence of events in authentication and an initial transaction with a server.

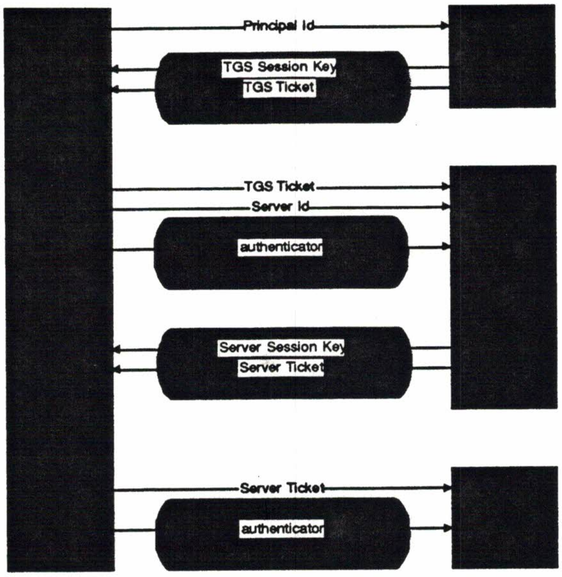


Figure 8

The steps involved in authentication of a human user are as follows:

1) The user runs the dcelogin command, passing it his principal id. dcelogin sends a message containing the principal id to the authentication server.
2) The Authentication Server does several things.
    a) It randomly generates a session key for the principal to use when accessing the Ticket Granting Server. But now it has a problem: how to inform both the client being authenticated and TGS of the session key.
    b) The Authentication Server creates a TGS ticket containing the principal's session key. It looks up the Ticket Granting Server's password (actually it's the TGS's current session key) and uses it to encrypt the ticket. Now only the TGS is able to interpret the ticket.
    c) The Authentication Server creates a packet of information containing both the TGS ticket and the TGS session key. It looks up the principal's password in the registry database (actually its a key derived from the password that's stored), and uses it to encrypt the entire package. It sends this package back to the client.
3) dcelogin prompts the client for his password and uses it to decrypt the package, obtaining the TGS session key that it will use and the TGS ticket that it will pass to the Ticket Granting Server. Note that both the client and the Authentication Server are protected by this exchange. If someone claims to be the principal but does not have the password, dcelogin can't decrypt the package. Conversely, suppose there were a impostor on the network claiming to be the Authentication Server. Since passwords are not transmitted to the server, the impostor can't collect them. Since the impostor doesn't have access to the registry database, it can't generate a packet that will be interpreted by the client.
4) Now suppose the user wants to access a particular server. (That's what it's all about!). The client program creates a special package, called an authenticator. The authenticator contains, among other things, the network IP address and the current time. The client encrypts the authenticator with the TGS session key that it received from the Authentication Server. It sends this encrypted authenticator to the Ticket Granting Server, along with the TGS ticket that it received from the Authentication Server (and which is encrypted in a key shared by the Authentication Server and the Ticket Granting Server). It also passes the identity of the server to which is wishes to connect.
5) The Ticket Granting Server decrypts the TGS ticket, which was encrypted using a key known only to the Authentication Server and the Ticket Granting Server. Among other things, it extracts the session key that is to be used by the client.
6) Using the extracted session key, the TGS decrypts the authenticator packet that had been created by the client and encrypted using the same key. Again, note that the client and the TGS are both protected. If

the client lacks the proper TGS session key, it can't generate a valid authenticator packet. The client cannot tamper with the TGS ticket that it received from the Authentication Server, since the TGS ticket is encrypted with a key that only the Authentication Server and the Ticket Granting Server know. Moreover, the client is protected from a Trojan horse TGS because the phony TGS will be unable to decrypt the TGS ticket it receives. Both are protected from a network eavesdropper who captures the ticket and modifies the return address because the authenticator contains the address in encrypted form.

7) The TGS generates a session key for the client to use with the application server. The session key is packaged in a ticket that is encrypted using the key that the application server shares with the TGS. This ticket and the application server session key are encrypted using the TGS session key that the client and the TGS share and returned to the server.
8) The client decrypts the package using the TGS session key and extracts the application server session key and the application server ticket.
9) The client creates an authenticator packet and encrypts it using the application session key that it has received. It sends the authenticator packet along with the application server ticket to the application server.
10) The application server decrypts the ticket that had been created by the TGS and encrypted with the key shared between the application server and the ticket granting server. It extracts the server session key and uses it to decrypt the authenticator.

Authentication of servers, as opposed to people, is slightly different. Passwords used by servers are stored in a file, called a *keytab* file, on the server's machine. These files are accessed by the server during its initialization. It is assumed that these files are sufficiently protected by local system security or that the server itself is physically secured.

For simplicity, the above discussion has ignored *groups*. Principals can belong to one or more groups. Group membership is recorded in the registry database. As part of the login transaction, the client receives a Privilege Attribute Certificate (PAC) that lists the groups to which its principal belongs. This PAC is passed to the Ticket Granting Server when the client requests access to an application server, and is incorporated into the server ticket. This allows servers to make authorization decisions based on group membership as well as individual identity.

One question you may be asking is how the registry database used by the Authentication and Ticket Granting Servers safely got the passwords of all the principals to begin with. They were entered using an administrative program. The one that ships with most current versions of DCE is called *rgy_edit*. rgy_edit is just another application program. It is used by a principal who logs on, is authenticated in the usual manner, and obtains a PAC from the Ticket Granting Server to access the Registry Service (i.e. server). Based on the verified identity and group memberships in the PAC, the Registry Server will, just like any other application, use the ACL facility to determine if the principal is authorized to modify the security database. The actual updates to the database, such as principal id and password, are transmitted as data in encrypted RPCs.

At first glance, all this might seem just to beg the question. How did the password of the principal updating the database get into the database, and how were his ACL privileges established to allow him to modify the database? Where did this all start? At the time the DCE cell is established, the cell initialization program defines an initial administrative principal, password, and appropriate ACL entries.

SECURITY AND REMOTE PROCEDURE CALLS

Seven security levels are available for remote procedure call:

- None
- Self assertion - Entities are who they say they are.
- Per connection authentication

- Per call authentication
- Per packet authentication
- Strong integrity check - The Data Encryption Standard (DES) checksum is used.
- Privacy - DES encryption is used.

The client and server negotiate the security level in effect for an interaction. The client proposes the security level to be used, while the server specifies the minimum security level it will accept.

Benchmarks indicate that, up through per call authentication, security has an insignificant impact on performance.

DISTRIBUTED TIME SERVICES

In an environment consisting of a single computer, the question of time is easy: it's whatever time the computer's clock says it is. In a distributed system, with multiple clocks, time becomes a much more complicated matter. Machines that interoperate require a common time for tasks like transaction logging and event sequencing. The DCE security services make extensive internal use of timestamps. A simple example of the need for a common time would be a make file that accessed files on multiple machines.

DCE's Distributed Time Service (DTS) loosely synchronizes the clocks in a cell with each other and optionally with an external time source, such as the National Institutes of Standards and Technology time service. DTS can also interoperate with the Internet Network Time Protocol. Time in DTS is represented, not as an absolute value, but as a value with an inaccuracy factor.

Each cell has one or more machines that run time server daemons. Periodically, each server adjusts its time based on the values of an external time source (which is definitive if available), its own system clock, and the times reported by other servers. Every machine in the cell not running a time server runs a time clerk daemon. Periodically, every clerk in the cell queries the server daemons for their time and inaccuracy factor and computes a common time. The DTS software clock on the individual machine is then gradually adjusted to bring it into agreement with the computed time. (In OS/2, this occurs at the rate of 100:1.) The hardware clock is also adjusted. (In OS/2, this occurs abruptly.)

THREADS

Thread services are, so to speak, the plumbing inside the plumbing. The Threads package allows for creation, management, and synchronization of multiple streams of execution within a single process. Since the threads of a process share a common address space, they can communicate easily. As a result, threads support parallel execution very efficiently.

Threads per se have nothing to do with networking. DCE contains the Threads package because DCE requires threads for its internal operation, and not all operating systems provide them. In OS/2, the threads package basically just repackages the native OS/2 threads facilities, while in other environments that do not provide native threads, such as many versions of UNIX, this is a full-blown subsystem. The threads package assures that all platforms on which DCE is implemented share a common threads interface.

In addition to their use internal to DCE, threads are available for use by DCE applications. Threads allow applications to exploit parallelism in several ways:

- Programs can execute concurrently on separate processors in a multiprocessor system.
- Servers can service multiple clients simultaneously.
- Clients can make multiple server requests simultaneously.

The DCE threads package conforms to the Posix 10003.4a (Pthreads) standard. It is based on the Common Multithread Architecture (CMA) from DEC.

DISTRIBUTED FILE SYSTEM

The Distributed File System (DFS) is a DCE application in the sense that it uses the DCE core facilities, but is not required in any way by those core facilities. It is, however, part of the DCE package and plays a central role in making the distributed computing environment transparent to end users. As DFS is not yet part of the OS/2 DCE implementation, we touch on it only briefly.

DFS is based on the Andrew File System from Carnegie-Melon and Transarc. It provides a single, unified namespace for all files, which is logically part of the cell namespace. File names are independent of file location, so files can physically be moved without changing their logical name. Access to files is controlled using Access Control Lists. The file system is journaled, and is designed for high performance. Files remain available to users even during administrative tasks such as backup. Collections of files, called *filesets*, can even be moved from one machine to another while the files are in use. Currently used files are cached locally for performance, transparently to the user. Replicas of commonly used files can be kept on multiple file servers. DFS can interoperate with the Network File System (NFS).

DCE RELEASES

As previously described, DCE is integrated on a "reference platform". This reference release is provided to vendors, who adapt it for their specific platforms and make products available in the marketplace. DCE has had the following reference releases:

1.0	January, 1991
1.0.1	August, 1992
1.0.2	May, 1993
1.0.3	December, 1993
1.1	November, 1994

Beware: Many vendors use a different numbering scheme for their distributions.

DCE FOR OS/2

The current (as of 4/95) release of DCE for OS/2 is based on DCE 1.0.2. It contains a Cell Directory Server, Security Server, and Time Server that run on OS/2 machines, and DCE client components for OS/2 and Windows clients. (A client here means any DCE component other than one of the servers.) Not included are the Distributed File System or Global Directory Services. This is essentially a "kick the wheels" release. IBM cautions that the server components are for development purposes only, and should not be relied on for production use. DCE execution performance is good, but memory residence requirements for this DCE release are considerable. Client machines should have at least 20 MB of RAM, and a machine running the full set of servers (Security, Directory, and Time) should have at least 24 MB.

Various versions of DCE have appeared on the Developer Connection CD-ROMs. Interestingly, The Developer Connection for LAN Systems Volume 2 (1/95), contains a stripped down DCE client that does not include security and simply synchronizes time when the client is started (which for LAN use is generally sufficient). This version has a radically smaller RAM footprint than the full DCE client, and is useful if all you need is the remote procedure call component of DCE.

In the second half of 1995, IBM plans to release DCE products based on OSF release 1.1. Unlike the current release intended for developers, these products should be of deployment level quality. Included will be major performance improvements, Workplace Shell cell administration, and the Distributed File System client capabilities. LAN Server Enterprise, which is currently in development and is intended for release in late 1995, is also expected to incorporate DCE facilities in an evolutionary manner.

RELATIONSHIP TO CORBA AND SOM

As previously noted, distributed objects represent another approach to application distribution. The relationship of DCE to distributed objects has been the subject of much discussion, debate, and standards negotiation, and a full review of the topic would fill another paper. The intermixture of technical issues with inter-corporate and intra-corporate politics is dizzying.

One view holds that DCE and distributed objects are complementary, that distributed objects should be layered on top of DCE facilities. Another sees them as fundamentally competing technologies addressing many of the same issues in application distribution.

The specification for distributed object interoperation that most of the industry has coalesced around is the Common Object Request Broker Architecture (CORBA) from the Object Management Group (OMG). Microsoft is the one notable exception, promoting instead its Common Object Model (COM). CORBA 1.0 defines only interface definitions, not implementation. Major products based on CORBA include IBM's System Object Model (SOM), HP's Distributed Object Management Facility (DOMF) and Sun's Distributed Objects Everywhere (DOE). CORBA 2.0, which is currently being defined, includes interoperability specifications. CORBA 2.0 requires that all implementations support a lightweight protocol (called Uno) as a General Inter-Orb Protocol (GIOP) for object request broker (ORB) interoperability. Vendors may optionally chose to implement an Environment Specific Inter-Orb Protocol (ESIOP), and HP, DEC, and IBM have indicated that they intend to use DCE in heterogeneous WAN environments. In such environments, CORBA implementations will likely also use DCE directory and security services

DCE and CORBA facilities overlap in many respects, and in fact their interface definition languages are quite similar. (Microsoft's COM interface definition language is a direct superset of DCE's.) Object proponents claim that application interfaces should be at the level of distributed objects, not at the level of remote procedure calls. They cite DCE's lack of polymorphism (though some rudimentary capabilities exist), inheritance, or the inability to discover interfaces (all interfaces to be used by a client must be specified in a DCE IDL file at compile time).

DCE proponents, on the other hand, point to the fact that DCE is a product that can be used today. Interoperability between ORB implementations is not yet possible, and since interoperability will be based on implementations derived from independent interpretation of the CORBA standard it is likely some way off. DCE, because its implementations derive from a common code base, can guarantee interoperability today. Whereas DCE has robust security and directory services, such facilities are just now being defined for CORBA.

The situation can be summarized by observing that DCE is a couple years ahead of distributed objects in its product life cycle. Some comparisons with IBM's DSOM may be instructive. Whereas DCE has comprehensive and reliable security facilities, in DSOM you are who you say you are. Whereas DCE allows marshaling of pointers to an arbitrary depth, DSOM provides only one level of indirection. And DSOM has only primitive directory services. These are not fundamental limitations of DSOM. They will eventually be addressed, perhaps by incorporating facilities from DCE, but this will take time.

STRENGTHS AND WEAKNESSES

DCE is a big system, designed for solving big problems. Although DCE encapsulates much of the complexity of distributed systems, it itself is formidably complex. In its current form DCE is overkill for small, homogeneous LANs. For loosely coupled systems, message oriented middleware may be more appropriate. Nor is DCE the solution of choice if all you are doing is operating within a single application domain such as a DBMS that provides its own proprietary mechanism for function distribution.

With over 400 API calls, DCE takes time for programmers to learn. Error messages are frequently cryptic, giving little clue as to what the actually problem. Because of their mutlithreaded implementation, DCE applications can be difficult to debug, though this difficulty should be alleviated as distributed, thread aware debuggers become available. On the other hand, remote procedure calls can often be replaced during development by local calls. This is an advantage that DCE development has over systems that use fundamentally different mechanisms for to access local and remote services.

Eventually, it's likely that most programmers will use DCE indirectly. Development tools are appearing that use DCE internally while shielding the programmer from the details of DCE. As is commonly the case, tool availability on OS/2 lags that for other platforms. Significant application development products that use DCE internally include Entera from Open Environment, OODCE from HP, and Connection/DCE from Open Horizon. DCE is also the basis of major tools like the Encina TP monitor and is being incorporated into others like CICS.

In addition to the organizational learning curve, maintaining a DCE cell entails significant administrative effort. Part of the problem is the current set of arcane and inconsistent command line administrative interfaces. This will be alleviated by the new unified control program (dcecp) of DCE 1.1 and a host of graphical tools for DCE administration that are appearing on the market.

In many ways, DCE evidences design choices that were made either implicitly because of its UNIX origins or explicitly because of the need to limit variability along at least some design axes if a product was to actually be built. In principle, DCE is transport neutral, but in practice it almost always uses TCP or UDP. In principle, it can use any authentication method, but in practice it must be Kerberos. In principle, it can use any transfer syntax, but in practice it must be NDR. In principle it can be used from any computer language, and there are language interfaces for COBOL (Microfocus), Ada, and Visual Basic (Gradient), but in practice DCE is very much C oriented.

In short, DCE is not a universal panacea for application distribution. However, for the heavyweight problems faced by large organizations deploying hundreds or thousands of systems across heterogeneous LANs and WANs, DCE provides a robust, currently available solution that radically simplifies program development and system maintenance. DCE will also see wide adoption as facilities like RPC are used transparently by application generators, while other facilities such as Security Services are incorporated into programs like LAN Server and made available for general application use.

FOR FURTHER INFORMATION

Distributed Computing Environment - Understanding the Concepts, IBM Publication GC09-1498.
Harold Lockhart, Jr., *OSF DCE: Guide to Writing Distributed Applications*, McGraw-Hill, 1994. ISBN 0-07-911481-4.
Robert Orfali & Dan Harkey, *The Client/Server Survival Guide with OS/2*, Van Nostrand Reinhold, 1994. ISBN 0-442-01798-7.
Ward Rosenberry, David Kenney, & Gerry Fisher, *Understanding DCE*, O'Reilly & Associates, 1992. ISBN 1-56592-005-8.
Bob Russell & Benetta Perry, "Distributed Performance Charactersitics of IBM DCE for OS/2," *IBM Personal*

Systems, July/August 1994, pp. 27-35.
John Shirley, *Guide to Writing DCE Applications*,O'Reilly & Associates, 1992. ISBN 1-56592-004-X.

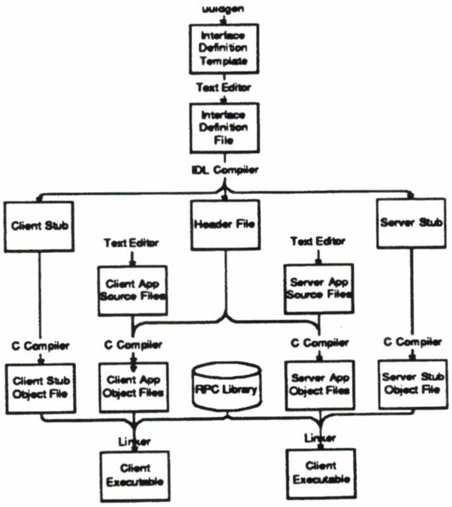
FRAGMENTS

DCE applications are split into client and server components. The client and server parts typically run on different machines.

Procedure calls are a well understood paradigm for managing program complexity. Programs are decomposed into layers, each layer providing a well-defined set of services to the layers above it, and relying on the services of those below it. Remote procedure calls extend this familiar and robust programming paradigm to the network environment.

APPLICATION DEVELOPMENT STEPS

- Generate Interface Identifier
- Write Interface Definition
- Run IDL compiler
- Write Client program
- Write Server Initialization code
- Write Server
- Compile and link client
- Compile and link server initialization and module
- Install server
- Run Client


uuidgen
Interface Definition Template
Text Editor
Interface Definition File
IDL Compiler
Client Stub
Header File
Server Stub
Text Editor
Text Editor
Client App Source Files
Server App Source Files
C Compiler
C Compiler
C Compiler
C Compiler
Client Stub Object File
Client App Object Files
RPC Library
Server App Object Files
Server Stub Object File
Linker
Linker
Client Executable
Client Executable

WHAT'S COMING IN 1.1 (OS/2 2.1)

Remote start
International character support
Aliases
Unified runtime daemons and control program (DCECP).

DFS

A comparison with other file systems may help to illustrate the power of DFS.

The design of NFS is optimized for tens of users. Files must be accessed with the MOUNT command, and there is no caching. Files have different names depending on how the file system they are part of is MOUNTed.

LAN Server and Netware 3.x are optimized for hundreds of users. Aliases provide location transparency. For example, one can issue the command NET USE V: DATA, where DATA represents a file system whose physical location can change.

By contrast, DFS is optimized for thousands of users.

. It will be possible to use DCE Security Services in place of the current LAN Server security. In addition, DCE Directory Services will be incorporated into DCE and it will be possible to manage a LAN Server domain as if it were a DCE cell and interoperate with other cells. <or will the LS Domain be part of a larger cell?>

TRADEMARKS

AT&T: GIS
Data General Corporation: AViion
Digital Equipment Corporation: VMS, Ultrix
Fujitsu: UXP/DS
Gradient Technologies:
Groupe Bull: GCOS
Hewlett-Packard Corporation: Apollo, HP, HP/UX, OODCE,
Ingres Corporation: INGRES
Informix, Inc.: Informix
IBM Corporation: APPN, AS/400, CICS, CPI-C, DSOM, IBM, LAN Server, MVS, OS/2, SOM, VM
International Institute of Electrical Engineers (IEEE): POSIX
Massachusetts Institute of Technology: Kerberos
Microsoft Corporation: Compound Object Model (COM), Object Linking and Embedding (OLE), Windows, Windows NT
Micro Focus, Inc.: Micro Focus COBOL
Novell, Inc.: NetWare, Novell, UnixWare
Object Management Group: CORBA
Open Software Foundation, Inc.: DCE, OSF, OSF/1
Oracle, Inc.: ORACLE
Peerlogic, Inc.: PIPES
Santa Crus Operations, Inc.: SCO UNIX
Siemens-Nixdorf: BS2000, SINIX
Sun Microsystems Inc.: ONC, RPC, SUN
Sunsoft, Inc.: Network File System, Solaris
Sybase, Inc.: Sybase
Tandem Computers: NonStop
Transarc Corporation: Andrew File System, Encina
UNIX Systems Laboratories, Inc.: Tuxedo, UNIX

MGT 0202

The New Face of Teamwork

Richard Sullivan
IBM
Somers, New York

UNLEASHING THE POWER OF TEAMWORK ... IT'S ABOUT COMPETING NOT COMPUTING.

In today's competitive marketplace the difference between winning and losing is determined by how quickly a company responds to its customers' needs—needs that frequently change. Responding to these changing needs requires new and innovative ways of managing your business. Companies throughout the world are implementing mobile work forces, flattened organizations, dynamic teaming, downsizing of the work force, and process reengineering, all in an effort to get to market faster, with a better product than their competitors. There is unprecedented pressure to ensure that business processes are streamlined to avoid unnecessary work, and that all resources (such as people, money, and materials) are aligned with business goals. In short, the right person must be doing the right work, with the right information, at the right time.

At the same time that these forces are at play in the business world, information technology is also advancing at an accelerating pace. It is not surprising then that this combination has caused an explosion of interest in workgroup computing, where computers are used to assist people in working together collaboratively to drive their business. Workgroup computing, as it is known today, has evolved from a heritage of related applications. First, individuals discovered the productivity benefits of using personal tools such as word processors, spreadsheets and graphics packages, and within a short time, their usage was pervasive. These workers quickly realized the need to share the information they were creating on their personal computers, first by simple file sharing, then through electronic mail, followed by shared databases and other information sharing capabilities.

No doubt technology has enabled **individuals** and **teams** to work more effectively, but now it is time to take the next step, to make the **overall business** more productive. In order to have the enterprise-wide impact needed, personal productivity, workgroup and line of business applications must be tied together and implemented so that they have the following characteristics:

- They must REFLECT THE WAY BUSINESSES AND PEOPLE ACTUALLY WORK (not the way the software works).
- They must be RELIABLE, SECURE AND DEPENDABLE, as they become increasingly critical to the mission of the business.
- They must FIT INTO EXISTING BUSINESS ENVIRONMENTS, protecting the investment that has been made in systems and networks.

The need is for a comprehensive portfolio of functions that have been designed and developed from their inception to deliver on these requirements: from communications facilities such as mail and directory, to information management capabilities such as search and multimedia libraries, to work automation functions, such as scheduling, forms routing and workflow, the tools needed to connect people and information across the enterprise.

PRODUCTION TO COLLABORATIVE ... THE SPECTRUM OF WORK

The first objective is to reflect the way businesses and people actually work. However, not everyone works the same way. Some people perform tasks that are highly structured, transaction oriented, repetitive and well defined. These are production oriented, and are usually critical to the day to day running of a business. For example, health insurance claims processing and accounts payable processing systems are typical examples of production work.

Other people work in a way that is less structured and not as well defined. Their tasks are typically non-repetitive, of longer duration and project oriented. Business planners and market analysts participating in projects such as product introduction or sales proposals are examples of this type of work. While not as production oriented, the knowledge of these workers, and the decisions which they are making collaboratively, are equally critical to the overall success of a business.

To date, most business process reengineering efforts have focused on production activities, resulting in significant productivity gains and cost savings for many companies. Now, the technology is available to improve the productivity of knowledge workers in collaborative teams as well.

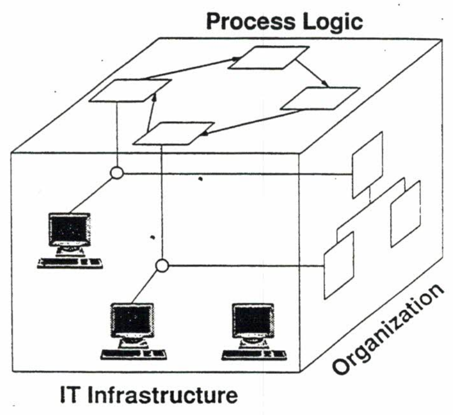
In order to truly affect the way your business operates, a workgroup solution must address all your information requirements, without distinguishing between personal productivity, collaborative and line of business applications. Only by encompassing the entire range of business activities can your workgroup computing solution deliver the productivity benefits needed to sustain a competitive advantage. Capabilities that facilitate communications, the management of information, and the automation of work need to be seamlessly integrated.

The need to communicate not only among people, but also among applications, has placed new and increased demands on existing communication systems. Electronic mail, which has primarily been used for simple text correspondence, is now being asked to handle the more diverse requirements of documents, images, spreadsheets, graphics, voice, video, and programs. Once a simple application, mail has evolved to the role of "middleware", where it provides the transport mechanism for more complex business communications. This type of communication (between applications) is called "messaging", and is used to facilitate activities such as forms routing, personal video conferencing, workflow, and key business processes such as customer support, product development and order processing. Today's requirements are for industrial strength messaging capabilities with functionally rich, multimedia E-mail, to ensure that communications occur reliably and easily across the enterprise.

The effectiveness of a business is also dramatically impacted by the degree to which information can be accessed for informed decision making. Information in today's organizations is stored in all manner of formats, such as spreadsheets and word processing files, voice messages, images and documents, as well as traditional structured data. This asset, the information contained in an organization, needs to be managed across the spectrum of disparate systems where it resides. Both structured (relational database) and unstructured (multimedia electronic information) need to be queried on an ad hoc basis regardless of where they reside. Application development tools are needed to allow existing applications (such as production line of business systems) to be easily integrated with new client server collaborative tools. The result is that critical business information is readily available upon demand.

Effective management of work within an organization means that activities happen in the right order, are assigned to the correct people, and carry along the necessary and correct information. The key to enterprise efficiency is to link the business activities, the skills of the people in the organization, and the information technology to support the activities. These three dimensions of effective work management are illustrated below:

Three Dimensions of Workflow



Software to manage the flow of work can be the catalyst that allows improvements to business operations to be made effectively and implemented consistently. Workflow capability must build on the communications and information management capabilities described above, so that desktop and line of business applications (on any existing platform) can be linked into a seamless work automation system. This approach reduces the opportunity for missed steps, incomplete data, and ensures that a company's most valuable resource, its people, work as effectively as possible.

The workflow capability should be implemented on a client server design that is built on a true object model allowing a business to continually monitor its activities, and make improvements based on feedback from the workflow manager. Bottlenecks can be readily identified, and changes can be made to reflect the dynamic needs of the business. The power of workflow will be expanded even further through the linkage with rich application development, simulation and business modeling tools.

INFRASTRUCTURE ... A FOUNDATION TO BUILD ON

The second objective is to provide reliable, secure and dependable offerings. Workgroup applications are no longer tools used only for personal productivity, but are becoming critical components of a business's operations, and as such must have industrial strength characteristics, based on offerings that have proven themselves under fire in production environments. For example, the same relational database that is commonly used for other "bet your business" applications. Messaging and queuing technology that is secure enough for banking and finance institutions to transport their monetary transactions and ensure fail-safe delivery of messages. Workgroup functions should be deployable at a department level and also scalable to large, complex networks where mission critical information is transmitted across and between enterprises.

As the Information Highway gains traffic and complexity, the need for reliability and dependability increases. Each day, additional commerce is taking place between enterprises in the form of electronic forms, electronic mail, complex multimedia documents, and information databases. The infrastructure must be up to the task of delivering those important business communications with the reliability, dependability and security that you expect.

INVESTMENT PROTECTION ... BUILD ON AND ENHANCE CURRENT INVESTMENTS

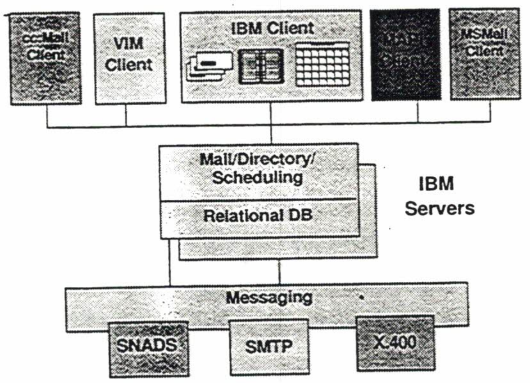
The third objective is to fit into existing business environments, protecting the investments made in systems and networks. Rare is the business today that does not have a multitude of differing systems in place. Significant investments have been made in hardware, operating systems, databases, office systems and business applications, and all for good reasons. For example, clients for Windows and OS/2 as well as Macintosh, Windows 95 and Motif. Servers for OS/2, OS/400 and AIX/6000, and additional platforms such as Hewlett Packard's HP/UX, Sun Microsystems' Solaris, Microsoft's Windows NT, Novell's NetWare and IBM's MVS/ESA, as driven by customer needs.

In addition to operating system issues, companies are also struggling with issues of interoperability between similar applications. For example, many businesses have multiple E-mail systems and document libraries installed. Gateways to address coexistence requirements are needed.

Other interoperability requirements are satisfied through industry standard APIs, such as Vendor Independent Messaging (VIM), Mail API (MAPI), Common Mail Call (CMC) and Calendaring and Scheduling API (CSA). This means that an investment in applications that implement these standards will not only be protected, but will be enhanced as those client applications connect to new robust server function.

A third way to satisfy coexistence requirements is through native communications protocols. Also required is support for various mail interchange standards, including SMTP/MIME, X.400, and SNADS.

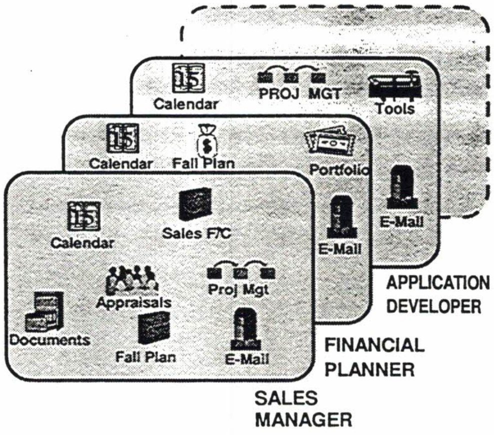
Enterprise Communications


cc:Mail Client
VIM Client
IBM Client
MAPI Client
MSMail Client
Mail/Directory/ Scheduling
Relational DB
IBM Servers
Messaging
SNADS
SMTP
X.400

DIRECTIONS ... WHERE WE GO FROM HERE

IBM's vision for workgroup computing is that the workers will never see a standalone product on their workstations. Sound strange? It shouldn't. The goal is to evolve to an environment where the user sees a desktop that reflects his or her job—and nothing else. The worker will click on an icon that represents a task, and the behind the scenes functions will lead the worker through the steps that are needed to complete the task. For example, a manager may see icons that represent personnel, major projects and financial results, as illustrated in the picture below. The information that is delivered to the manager's desk may actually have used mail, scheduling, workflow, or multimedia library, but the manager need never be concerned with starting the applications or navigating from one to another.

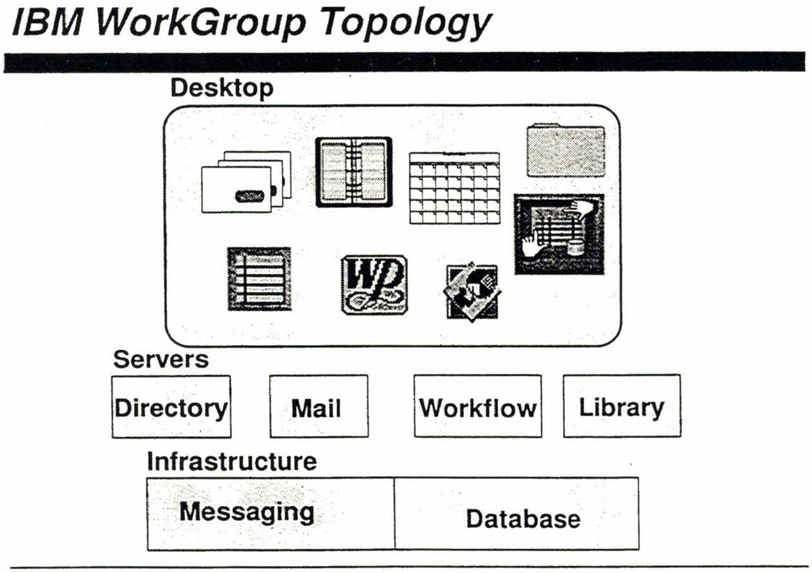
Custom Desktops


Calendar
PROJ MGT
Tools
E-Mail
APPLICATION DEVELOPER
Calendar
Fall Plan
Portfolio
E-Mail
FINANCIAL PLANNER
Calendar
Sales F/C
Appraisals
Proj Mgt
Documents
Fall Plan
E-Mail
SALES MANAGER

Object technology makes the scenario possible. IBM is well underway in the creation of OpenDoc objects that will be used in the IBM WorkGroup portfolio, as well as delivered as part of a solution library. Through this library, IS professionals and systems integrators can build highly customized solutions that meet a business's unique needs. Visual application development tools make this approach easy for both professional developers and sophisticated users

INTEGRATION ... THE WHOLE IS GREATER THAN THE SUM OF THE PARTS

Today, many workers spend valuable time navigating between applications instead of applying their knowledge to their jobs. Users are spending their time learning various tools, and passing data between them. Systems administrators are busy adding, deleting and changing enrollment information about users, installing multiple applications and providing interfaces between them. The IBM WorkGroup approach is to provide a well integrated set of functions that save both these communities valuable time.



IBM's strategy is to provide core functions that are used and reused across the IBM WorkGroup portfolio. For example, messaging will be used not only for delivering E-mail messages, but also for delivering task assignments coordinated by the workflow function. The key capabilities being developed will be used not only within the IBM portfolio, but through industry standards will also be open for customization and extension by independent software vendors, value added resellers, service organizations and I/S professionals.

The introduction of IBM WorkGroup capabilities delivers on the integration objective. Users are provided access to mail, directory, scheduling and calendaring, facsimile and agents via a single logon. They will experience a common look and feel across the portfolio of function, including a common tool bar, colors, fonts and operations. Administrators will have a single enrollment and user management facility for use not only with this set of IBM WorkGroup facilities, but also with other popular industry applications as well. Additional functions such as electronic forms, personal conferencing and workflow will also be integrated at the desktop, and the underlying infrastructure will continue to be integrated at the server level. For example, the messaging and relational database components will be "under the covers."

The goal is increased simplicity, with functionally rich, cohesive offerings. They will share common installation and administration, and will have the same look and feel. They will easily integrate not only with each other, but will also provide open interfaces and links to other vendors' popular products. Through integration, IBM WorkGroup delivers on the promise that the whole is indeed greater than the sum of the parts..

The huge investment that many companies have in desktop documents is also protected. The library facility embraces the popular desktop productivity applications. In fact, documents of any type (text, image, audio, video, or graphics) can be stored and cataloged. This means that users can create and edit documents with the tool of their choice (such as a word processor or a spreadsheet), and have a common way of storing, searching and retrieving them. This is particularly valuable for teams, where multiple people are working together and sharing documents. This approach not only protects the investment that has been made, but enhances it by adding document management capability.

SUMMARY ... HOW TO GET STARTED

IBM WorkGroup offers the ability to support the range of activities taking place in your business, with most of the function available through IBM products today. The IBM WorkGroup offerings implement the client server model, providing maximum flexibility for your company. Over the last two years significant additions have been made to the IBM portfolio of client server workgroup offerings: scheduling and calendaring (Time and Place); workflow (FlowMark); forms (FormTalk); document management (VisualInfo and Visual Document Library); and decision support, data access and business analysis tools (Visualizer). In November, 1994, the IBM WorkGroup portfolio was extended with the introduction of new client/server mail and messaging functionality. This continues the IBM tradition of employing technology to deliver rich end user function based upon a reliable, robust infrastructure.

In order to be ready for the future, we recommend that businesses establish a firm foundation. It is imperative that the underlying infrastructure of your workgroup computing environment support the demanding needs of the enterprise. Otherwise, you may find your company bumping up against the limitations of an implementation that was not built from the inside out. This is somewhat analogous to designing the body of a sports car and then trying to determine how to build an engine, transmission, and suspension that will fit inside the design.

By utilizing IBM WorkGroup, you can be assured that the proper foundation is firmly in place. The integration provided for mail, directory, scheduling and calendaring, facsimile, and agent functions will provide immediate productivity benefits for your users. Your system administrators' jobs will be made easier through simplified installation and enrollment capabilities.

An area from which your business can gain immediate value is document management. Extensive search capabilities, shared library facilities, and capabilities such as version control and check-in/check-out ease the management of your information. The information management capability of IBM WorkGroup will build upon IBM's Visual Document Library, which is available today to provide your teams easy access to the investment they have made in desktop produced documents.

Business processes that rely on existing line of business applications, LAN and desktop tools can be tied together with IBM's FlowMark. IBM consulting and services teams are now aligned by industry and stand ready with the skills and experience to work with you toward a successful workflow solution for your business.

IBM is committed to delivering solutions that help you improve the bottom line of your business. The investments that are being made in IBM WorkGroup cover a spectrum that ranges from the automation of complex production processes, all the way to the work efforts of a small collaborative workgroup. These functions are delivered on multiple platforms, and are built upon the robust infrastructure that is required to deliver the dependability you need. We recognize that any capability you decide to entrust to us must work with investments you have already made in products from IBM, and from our competitors. The past few years have seen unparalleled change in our industry, and we are excited about the technology that is evolving to enable the new face of teamwork. We look forward to continuing our partnership with you in this new era.

MM 0208

The OS/2 Multimedia Experience

Maria Ingold
IBM

Agenda

- Discuss multimedia in general
- Discuss features of OS/2 Warp mutimedia products
- High level look at how it all works
- Demonstrate OS/2 Warp multimedia products
- The future of OS/2 Warp multimedia
- Available audio and video cards
- Useful publications
- Available forums
- Download sites
- Useful phone numbers
- Other sessions of interest

What is Multimedia Used For?

- Games
  - Audio sound effects, dialog, and background music
  - Bitmap images for each or parts of scenes
  - Animation of actions, introduction, level changes
  - Video augmentation for some scenes
  - Text for hints, dialog
- Education/Presentations
  - Can listen to audio instead of/in addition to reading
  - Interact with image, animation, and video
- Communication
  - Video conferencing

Multimedia in General

- Audio
  - Digital Wave Audio
  - MIDI
- Image
  - Vector
  - Raster (Bitmap)
- Animation
  - 2-D
  - 3-D
- Video
  - Hardware Motion Video
  - Software Motion Video
- Text

Hardware Considerations

- Input of "real world" *analog* into computer *digital* form
  - Audio
    - Sound card and microphone or line-in input
  - Image
    - Scanners and cards to bring in 2-D images
    - Video camera and frame grabber card for 3-D images
  - Video
    - Video camera and frame grabber card
    - Video cards for hardware assisted video
- Conversion from computer digital form back to real world
  - Audio
    - Sound card and speakers, headphones, or line-out
  - Images
    - Printer

OS/2 Warp Multimedia Apps

- In the box...
  - Audio
    - Digital Audio Player/Recorder/Editor
    - CD Player
    - MIDI Player
    - System Sounds
    - Lotus/Excel Macros
  - Image
    - Multimedia Viewer (Bonus Pak)
  - Animation
    - FLI/FLC player
  - Video
    - Digital Video Player
    - Video In Recorder (Bonus Pak)
    - AVI File Utility (Bonus Pak)
  - Multimedia Data Converter

OS/2 Warp Multimedia Apps

- Shareware/Freeware Applications
  - PM Sound Exchange
  - Digital Music Player
  - Aquanaut
  - Roids
  - Sorry
  - PM Poze
  - Light Waves
  - and others...

Digital Audio

- Player/Recorder/Editor
  - Player features
    - Plays RIFF Wave (PCM), ADPCM
    - Plays specified ranges
    - Supports drag and drop
  - Recorder features
    - Stereo/Mono (2 channels/1 channel)
    - 11.025, 22.05, 44.1 kHz (Voice, Music, HIFi)
  - Editor features
    - Special effects - Mix, Fade, Reverb, Reverse, Echo...
    - Cut, copy, paste entire file or selected regions
    - Audio quality conversion

Sizes for Digital Wave Samples

Channels	Rate	Resolution	1 Sec(KB)	1 Min(MB)
1	11.025	8 bit	10.77	.63
1	11.025	16 bit	21.5	1.26
2	11.025	8 bit	21.5	1.26
2	11.025	16 bit	43	2.5
1	22.05	8 bit	21.5	1.26
1	22.05	16 bit	43	2.5
2	22.05	8 bit	43	2.5
2	22.05	16 bit	86	5
1	44.1	8 bit	43	2.5
1	44.1	16 bit	86	5
2	44.1	8 bit	86	5
2	44.1	16 bit	172	10

CD Player

- Displays title, track, time into track
- Automatic play, repeat, shuffle
- Play, cue, skip tracks, eject
- "No CD" graphic
- Digital transfer to audio adapter

MIDI Player

- Play back MIDI files from a selected position
- Generic media player

System Sounds

- Associate digital wave files with system events
- Can associate sounds created with OS/2 Tune Editor
- Can set volume for all sounds or individual sounds

Lotus/Excel Macros

- 16 bit and 32 bit versions
- Adds "Audio" option to menu bar
  - Play
  - Record
    - Graphical recorder
    - From microphone
    - Paste from clipboard

Multimedia Viewer

- View Images
- Play Movies/Animation
- Play Audio
- View Text
- Create Light Table thumbnails (look like photo slides)
- Use OS/2 standard apps by double-clicking inside frame
- Use other applications by double-clicking outside
- Uses drag and drop
- Can create Light Table References
- Adds extra image and audio formats

FLC/FLI Animation Player

- Supports playback of Autodesk FLC/FLI animation files
  - FLI
    - Up to 320x240
  - FLC
    - Up to 640x480
- Average frame rate of 15 FPS
- Wave file with the same name as FLI/FLC file plays along

Digital Video Player

- Software motion video CODECs
  - Ultimotion
  - Indeo 2.1, 3.1, 3.2
- Completely scaleable
- Colors are scaled to the capability of playback system
- Frame rate is also scalable

Video In Recorder

- Real-time recording
  - Video casette player, video recorder
- Frame-step recordirg
  - Frame-steppable laserdisk player
- Frame Grabber card
- Can set frame size, frame rate, data rate, and quality
- Compression types
  - Ultimotion
  - Indeo 2.1, 3.1, 3.2
- Can set contrast, brightness, color, and tint
- Monitor video and audio input
- Editing
  - Cut, copy, paste
- Convert between different video types
- Bitmap capture
- Can compress raw movies

AVI File Utility

- Browse Audio Video Interleaved (AVI) file information
  - Header, chunk ID and length, index, and status
- Change the interleave ratio of audio and video chunks
  - Ideal is 1:1
- Split audio and video apart
- Merge audio and video together
- Change the skew between audio and video
- Create "raw" animations from a series of images
  - Any type of OS/2 Mulltimedia image type
  - Equal size bitmaps
  - Can specify pauses
  - Can specify frame rate
  - Can compress using Video In Recorder

Multimedia Data Converter

- Convert between different audio types
- Convert between different image types
  - Can preview image files
- Can convert single file or multiple files
- Can select files of only specific types

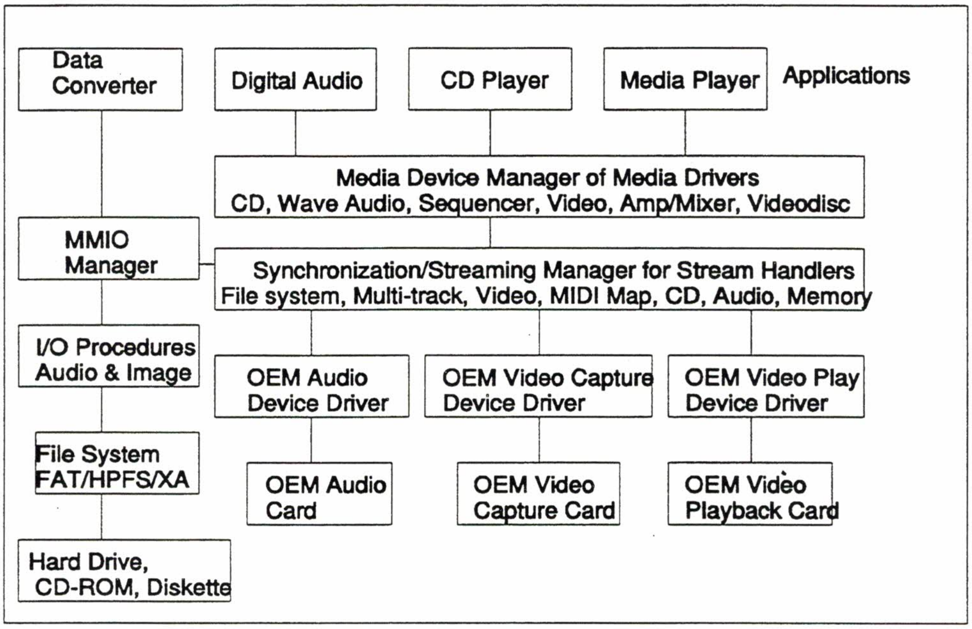
Audio Formats

■ Apple IFF Wave Digital Audio	.AIF
■ Amiga IFF Format	.IFF
■ Creative Labs Inc. Voice Audio	.VOC
■ IBM AVC ADPCM Digital Audio	._AU with ._AD
■ RIFF Waveform Digital Audio	.WAV
■ Unix (NeXT/SUN) SND Format	.AU, .SND
■ Musical Instrument Digital Interface (MIDI)	.MID

Image Formats

■ GIF Image Compressed	.GIF
■ IBM AVC Still Video Image	._IM, .!IM, and ._ID
■ IBM MMotion Still Video Image	.VID
■ IBM OS/2 1.3 Bitmap	.BMP
■ IBM OS/2 2.0 Bitmap	.BMP
■ Kodak PhotoCD	.PCD
■ MS Windows Device-Independent Bitmap (DIB)	.DIB
■ PCX (PC Paintbrush) Image Compressed	.PCX
■ RIFF Device-Independent Bitmap (RDIB)	.RDI
■ Targa Image Compressed	.TGA
■ Targa Image Uncompressed	.TGA
■ TIFF FAX Compressed	.TIF
■ TIFF Intel Compressed	.TIF
■ TIFF Intel Uncompressed	.TIF
■ TIFF Motorola Compressed	.TIF
■ TIFF Motorola Uncompressed	.TIF

High Level Overview of Multimedia



Data Converter

Digital Audio

CD Player

Media Player

Applications

Media Device Manager of Media Drivers
CD, Wave Audio, Sequencer, Video, Amp/Mixer, Videodisc

MMIO Manager

Synchronization/Streaming Manager for Stream Handlers
File system, Multi-track, Video, MIDI Map, CD, Audio, Memory

I/O Procedures Audio & Image

OEM Audio Device Driver

OEM Video Capture Device Driver

OEM Video Play Device Driver

File System FAT/HPFS/XA

OEM Audio Card

OEM Video Capture Card

OEM Video Playback Card

Hard Drive, CD-ROM, Diskette

Demonstrations

Audio Cards

- Ultrasound Ace by Advanced Gravis
- Ultrasound by Advanced Gravis
- Ultrasound Max by Advanced Gravis
- Sound Galaxy Nova 16 Extra by Aztech Systems Ltd.
- Sound Blaster (16, Pro, AWE-32) by Creative Labs
- Portable Sound Plus by DSP Solutions
- AudioDrive by ESS Technology Inc.
- Audiovation by IBM
- M-Audio Capture and Playback Adaptor by IBM
- Business Audio (AD1848 Chip) by IBM
- Sound Man 16 by Logitech
- Pro Audio Spectrum 16 by Media Vision
- Pro Audio Studio by Media Vision
- Jazz 16 by Media Vision
- WinSound by Sigma Designs

Audio Cards - Continued

- AD 1848 (Business Audio) by Analog Devices
- Plug-N-Play by Arkay Technologies
- CS4321/CS4248 by Crystal Semiconductor
- Audio Drive 488 by ESS Technologiy
- Audio Drive 688 by ESS Technology
- ISP 16 by Media Magic
- SC500 by Reveal
- Sound Expression 14.4 VSp by Boca Research
- Mozart 16 by Oak

Video Cards - Capture

- Video Capture Adapter by IBM
- SuperVIA by Jovian Logic
- QuickVIA by Jovian Logic
- M&M Basic by Omnicorp Graphics Corporation
- Super Video Windows by New Media Graphics
- Video Blaster by Creative Labs
- Video Blaster FS200 by Creative Labs
- Video Clipper by CEI
- Video Magic by Samsung
- WaveWatcher by AITech
- WinMovie by Sigma Design
- Illuminator-16 by Matrox
- Marvel II by Matrox
- MediaBurst Movie by IBM
- V7320 Video Controller by ASCII Corporation

Video Cards - Playback

- Video
  - V7320 Video Controller by ASCII Corporation
  - ReelMagic by Sigma Designs
  - ReelMagic Lite by Sigma Designs
- Television
  - Win/TV by Hauppauge
  - Win/TV Celebrity by Hauppage
  - PC/TV by Tecmar

OS/2 Multimedia Publications

- Articles
  - OS/2 2.1 Multimedia: From In-the-Box to Video In
    - Personal Software Technical Newsletter, #2
  - OS/2 2.1 Multimedia
    - OS/2 Monthly, Issue 10
  - Audio Demistified
    - OS/2 Magazine, September 1994

OS/2 Multimedia Publications

- Chapters
  - Ch 15, Multimedia and OS/2
    - Dvorak's Guide to OS/2
  - Ch 15, Multimedia
    - OS/2 2.11 Unleashed
- Books
  - The OS/2 Multimedia Advantage
    - S71G-2220
  - Ultimedia Video In User's Guide
    - S82F-4224
  - Can order the above books in US: (800) 342-6672
    - In Canada: (800) 465-4234

OS/2 Multimedia Forums

- Internet
  - comp.os.os2.multimedia
  - comp.multimedia
- America Online
  - keyword OS2
- Compuserve
  - OS2SUPPORT (Various sections)
  - OS2USER (Various sections)
  - OS2DF1 FORUM, Section 12
  - GO IBM
  - FIND OS2
- Prodigy
  - jump OS/2 CLUB
- World Wide Web
  - http://www.austin.ibm.com/pspinfo/os2.html

Download Sites

- America Online
  - keyword OS2
- Compuserve
  - OS2DF1, Section 12
- Prodigy
  - jump OS/2 CLUB
- FTP
  - hobbes.nmsu.edu
  - ftp-os2.cdrom.com
    - /pub/os2/32bit/multimedia

BBSs - USA

■ Fernwood	(203) 483-0348
■ The Bin BBS	(206) 451-1905
■ OS/2 Source BBS	(303) 744-0373
■ Denver OS/2 BBS	(303) 755-6859
■ Inside Technologies BBS	(313) 283-1511
■ OS/2 Woodmeister	(314) 446-0016
■ Gateway/2	(314) 554-9313
■ Pyramid/2	(415) 494-7497
■ Bay Area OS/2	(510) 657-7948
■ OS/2 San Diego	(619) 558-9475
■ OS/2 Las Vegas	(702) 433-5535
■ OS/2 Shareware	(703) 385-4325
■ Greater Chicago Online	(708) 895-4042
■ OS/2 Exchange BBS	(904) 739-2445

■ Information about US OS/2 BBSs: (800) 547-1283

BBSs - Europe, Australia, Canada

■ Abaforum (Barcelona)	34-3-589.38.88
■ IBM UK	44-256-336655
■ OS/2 UK	44-454-633197
■ IBM Denmark	45-42-88-72-22
■ Copenhagen UG BBS	45-48-24-45-80
■ OS/2 Norway	47-66-99-94-50
■ IBM Norway	47-66-99-94-50
■ IBM Germany	49-7034-15-2160
■ OS/2 Australia	61-2-241-2466
■ Soft/2 Shareware (Adelaide)	61-8-370-7339
■ IBM Canada	(416) 946-4244
■ IBM Canada	(416) 946-4255
■ IBM Canada	(514) 938-3022
■ IBM Canada	(604) 380-5411
■ IBM Canada	(604) 664-6464

Useful Phone Numbers

■ Audio Card Manufacturers	
• Advanced Gravis	BBS (604) 431-5927
• Arkay Technologies	Tel (603) 425-2149
• Aztech Systems Ltd.	BBS (501) 623-8933
• DSP Solutions	BBS (415) 494-1621
• Media Vision	BBS (510) 770-0527
■ Video Cards	
• Matrox	Tel (514) 685-2630
• Omnicomp Graphics	Tel (514) 685-2630
• Tecmar	Tel (800) 422-2587
■ IBM	
• (800) 3 IBM OS2 (3 426 672)	

Other Sessions of Interest

- For Users
  - OPPS11 Harnessing the Power of the Warp BonusPak
  - DDOS28 OS/2 Multimedia User Tips and Techniques
- For Developers
  - MMOS02 OS/2 Multimedia for Developers
  - DDOS18 Video Playback and Capture Under OS/2
  - DDOS24 Multimedia DD Overview and Future Directions
  - DDOS25 Multimedia Audio Device Drivers
  - DDOS26 Multimedia Video Capture Device Drivers
  - DDOS27 MPEG Playback Device Drivers
  - DDOS29 Multimedia Device Driver Installation and Test
  - PPOS05 Natural Computing on OS/2 for the PowerPC

PRO 0203
SOM Programming on OS/2

Harri Madduri
IBM
Austin, Texas

1.0 INTRODUCTION

Object technology is believed to make life easier by promoting reusable software components, i.e., objects. Theoretically, because objects are reusable, writing new applications becomes progressively easier as we build more and more of them (since we accumulate repositories of reusable objects). What aspects of object technology might make such a vision possible? One is encapsulation — the principle of not exposing the internal (implementation) details of an object. Code written following this principle lives longer than code written otherwise. The other aspect of object technology that helps reusability is inheritance. Through inheritance, object implementations can be refined over time as needed. One can reuse all functions (i.e., behaviors) of the original object that are still usable and override (or redefine) only those that need to be different. Using this very powerful mechanism, we only need to write code for new object types and that only for ways in which the new types differ from the existing ones.

All of this sounds wonderful. If this is true, then by now we should have seen at least hundreds if not thousands of commercial object libraries (actually class libraries). And, writing new software in many cases should be as simple as picking up a few good objects from different libraries (perhaps from different vendors) and gluing them together with minimal amounts of code. The reality is far from this objects dream. Let us examine the reasons.

The reasons are several. First, there are multiple object-oriented languages, and each has its own object model. Each offers the benefits of object technology through class libraries and hence offers reuse. However, the class libraries built in different languages do not interoperate. For example, if there is a "Queue" class built in Smalltalk and if I am a C++ programmer, then I cannot subclass (i.e., specialize) Queue in my C++ program to produce say a "Priority Queue". Thus the available class libraries are limited by one's choice of language. Contrast this with procedure-call libraries like math libraries that can be called from multiple languages and hence lead to a high degree of code reuse. The problem with class libraries is that there are no binary standards unlike for procedure-call libraries.

Second, there is an interoperability problem not only among different object-oriented languages, but also within the same language among different compilers. For example, in general, developers cannot use a C++ class library that was built using a different C++ compiler than their own (unless, of course, the class library was obtained in source form). The same is true of other object-oriented programming languages. Thus, to reap the benefits of object technology, class-library vendors must ship their libraries either in source form or in binary form for a specific compiler.

As can be guessed, there are serious inhibitors for shipping class libraries in source form. Building a good class library is much harder than usual object-oriented programming, since the library is meant to be used in a variety of applications, not just one. Also, giving away source might imply a short-lived market for the library, since "new and improved" libraries based on the original can be produced by competitors very quickly and easily. If a vendor must make money by selling only a few copies of the class library then the copies would be very expensive, which inhibits a market for class libraries. Shipping class libraries in binary form is being done today. In general, the developers are limited to the libraries provided by their compiler vendor.

Third, there is a whole set of problems (especially with C++-like languages) with regard to maintaining upward compatibility across different versions of a binary class library. For all the above reasons, we believe that the promise of object technology has not been fulfilled.

This rest of this paper briefly describes how IBM object technology is beginning to make reuse a reality through its System Object Model (SOM), and how its extension to a distributed environment (DSOM) makes distributed object-oriented programming easier.

2.0 SYSTEM OBJECT MODEL (SOM)

SOM is a packaging technology for class libraries. The main characteristics of SOM are:

o Language neutrality,
o Binary upward compatibility (for industrial-strength libraries), and
o Flexible and efficient (state-of-the-art) object model.

SOM provides a language-neutral object model that has elements of both static and dynamic object models (e.g., C++, Smalltalk, and Objective-C). Although SOM is not strictly a language, it includes syntax for expressing class definitions and a compiler that reads these definitions and generates the appropriate language bindings. Language bindings are language-specific header files that allow an object to be used (or implemented) in a given language. The syntax for expressing class definitions has been adopted from the Object Management Group's CORBA (Common Object Request Broker Architecture) standard. This standard is called the interface definition language (IDL). Thus, all SOM objects are defined in an industry-standard language and can be implemented in any language of the implementor's choice (assuming the bindings for that language are avail able).

SOM objects can be built (i.e., implemented) in one language and used from a different language. The use of SOM is not limited to object-oriented languages. SOM objects can be used even from non-object oriented languages, this feature allows us to wrap existing non-object oriented code into SOM objects.

SOM libraries support binary upward compatibility across releases(i.e., preserve binary compatibility). There are several changes that the provider of a SOM superclass can make without requiring that subclasses be recompiled. By contrast in C++, changes like adding data members or moving a member function up the hierarchy in a super class forces recompilation of its subclasses. Not so in SOM. This binary compatibility is key to the success of a software component market. Without such a packaging technology, each application may require its own version of the class libraries, even if the interfaces to the class libraries haven't changed. Moreover, if a single application uses objects based on different versions of a class library, this could pose significant technical problems, besides requiring the end users to buy much more memory to accommodate multiple versions.

SOM provides a state-of-the-art object model. In SOM, classes are themselves objects. This feature of SOM allows tremendous flexibility in managing instances of a class and in controlling the behavior of instances at run time (compare this to a static language like C++). Yet, SOM retains much of the efficiency of C++-like compiled languages. The SOM kernel supports three types of method resolution: method offset (like C++), name lookup (like Smalltalk), and dispatch-function resolution. The last type of resolution is accomplished through a special method called somDispatch, which offers excellent capabilities to customize the behavior of any class instances. For example, this feature is exploited in DSOM to extend SOM to a distributed environment. This feature can also be exploited to make a class whose instances are all thread-safe. (Hint: somDispatch allows applications to intercept every method call on every instance of the class.) This also provides an attractive feature: thread-safety can be built into a metaclass and thereby applied to as many classes as desired simply by declaring them to use that metaclass. This kind of metaclass programming represents the best value possible paper. (An interested reader is referred to the SOMObjects Developer Toolkit User's Guide.) Just to give a flavor of programming SOM libraries, I show a simple example in the next subsection.

2.1 AN EXAMPLE PROGRAM

```
                                  | #include <somobj.idl>                              | |
| | interface Stack: SOMObject {                          | | // Constants
| |   const long stackSize = 100;                         | | // Methods
| |   boolean full();                                     | |   boolean empty();
| |   long top();                                         | |   void pop();
| |   void push(in long el);                              | | #ifdef __SOMIDL__
| |   implementation {                                    | |     releaseorder: full, empty, top, pop,
push;          | |        dllname = "stack.dll";                        | |      // Method modifiers
| |     somInit: override;                                | |     // Instance data
| |     long stackimpl[stackSize];                        | |   };
| | #endif                                                | | };
|                                     **** BOLDEN AS SHOWN IN FOIL #11 (please print OS2World95.ps)
```

Figure 1: IDL definition for the Stack class

The keyword "interface" in CORBA IDL is the equivalent of "class". The above definition shows five methods defined on a stack object (full, empty, top, push and pop). SOM's extension to CORBA IDL also allows a developer to specify some implementation-specific information in the IDL file. This information is guarded by an #ifdef __SOMIDL__ statement. Note that the instance data (stackimpl array) is also in the implementation part.

Figure 2 shows the process of building a SOM class library. For example, to build a stack class and make it a SOM class library, we produce the file stack.idl and then run the SOM Compiler on it to produce the implementation and usage bindings.

```
                                  |                                                  | |
| |                                                 | |
| |                                                 | |
| |                                                 | |
| |                                                 | |
| |                                                 | |
| |                                                 | |
| |                                                 | |
|

**** Diagram only (no heading) from foil #9 of my talk. Please print OS2World95.ps
```

Figure 2: SOM Library Development Process

The SOM compiler also generates an implementation skeleton, stack.c that is used to build the stack class. We fill in the skeleton with appropriate code using any text editor and then compile it with the implementation binding file, stack.ih. The compiler (corresponding to the chosen implementation language) compiles and produces a dynamic load library, stack.dll. Now any client program wanting to use instances of stack or wanting to subclass from stack will use this dll. Any application that uses stack.dll can upgrade to later versions of stack.dll by simply replacing the dll, and without having to recompile the code.

```
                                  |                                                  | |
| |                                                 | |
| |                                                 | |
| |                                                 | |
| |                                                 | |
| |                                                 | |
| |                                                 | |
|                                  **** Diagram only (no heading) from foil #10 of my talk. Please print
OS2World95.ps
```

Figure 3: SOM Client Development Process

Figure 3 shows how to build a client program that uses the stack class from stack.dll. The usage binding file stack.xh (for C++ clients) is produced from the IDL definition by the SOM Compiler. The client's main program, together with this binding file and a library definition file stack.lib, can be used to produce a client executable. When this executable is run, stack.dll is loaded by the SOM kernel automatically when needed.

```
 ------------------------------------ | #include "stack.xh"                                   | |
 |  | int main() {                                     | |   Stack *stk;
 | |   Environment *ev;                                | |
 |  |   ev = SOM_CreateLocalEnvironment();              | |   stk = new Stack;
 | |                                                  | |  stk->push(ev, 100);
 | |   stk->push(ev, 200);                            | |  stk->pop(ev);
 | |   if (!stk->empty(ev))                           | |    somPrintf("Top: %d\n", stk->top(ev));
 | |        delete stk;                                   | |
 | |   SOM_DestroyLocalEnvironment(ev);               | |  return(0);
 | | }                                                |  ------------------------------------ **** BOLDEN AS
SHOWN IN FOIL #12 (please print OS2World95.ps)
```

Figure 4: Stack Client Program main.C

Figure 4 shows the client main program that creates an instance of stack (i.e., pooh) and invokes methods on it. The important point here is that the client is unaware of the language in which stack was originally built. It could have been C, or C++, or some other language. The binding file stack.xh makes it appear that stack was also implemented in C++.

2.2 DISTRIBUTED SYSTEM OBJECT MODEL (DSOM)

DSOM extends SOM to multiple processes and machines. Using DSOM, a SOM object can talk to another SOM object regardless of whether the two are located in the same or different processes (possibly on different machines). Method calls on a target SOM object always look exactly the same whether the object is local or remote. Thus DSOM provides location-transparent invocation on objects. This transparency is achieved as follows: For every remote SOM object that needs to be talked to DSOM creates a local "proxy object" that supports exactly the same interface as the remote object. All method calls on the proxy object are transmitted to the real object and the results (if any) are then brought back and returned to the caller. As alluded to earlier, the proxy uses the somDispatch mechanism to forward calls to the remote object. To the caller, the proxy looks like the actual remote object.

DSOM is standards-based. In fact, it is one of the most fully CORBA 1.1. compliant implementations. SOM, DSOM and several SOM-based class libraries are part of the SOMObjects Developer Toolkit product from IBM. This IBM product currently runs on AIX, OS/2, and Windows.

```
 ------------------------------------ | #include <somd.xh>                                         | | #include
"stack.xh"                                 | |
 |  | int main() {                                     | |        ...
 |  |   ev = SOM_CreateLocalEnvironment();              | |   SOMD_Init(ev);
 | |   stk = SOMD_ObjectMgr->somdNewObject(ev, "Stack","");| |
 | |   stk->push(ev, 100);                            | |  stk->push(ev, 200);
 | |   stk->pop(ev);                                  | |  if (!stk->empty(ev))
 | |     somPrintf("Top: %d\n", stk->top(ev));         | |  SOMD_ObjectMgr->somdDestroyObject(ev, stk);
 | |   SOMD_Uninit(ev);                               | |
 | |   SOM_DestroyLocalEnvironment(ev);               | |  return(0);
 | | }                                                |  ------------------------------------ **** BOLDEN AS
SHOWN IN FOIL  #14 (please print OS2World95.ps)
```

Figure 5: Client Program of a Remote Stack

Figure 5 shows a client program that is accessing a remote stack object. As can be seen in comparison to figure 4, the only differences are in creating a remote stack and destroying a remote stack. Once a stack is created, methods are invoked on it exactly as in the local case. SOMD_Init and SOMD_Uninit are functions to initialize and unInitialize the DSOM environment and these only need to be done once in the program. Programmers can write largely local/remote transparent programs by limiting object creation/destruction to a small set of modules and using the object references in the rest of the modules.

3.0 DISTRIBUTED SOM AND DISTRIBUTED OBJECT COMPUTING

The challenges faced by distributed procedural computing are also the ones faced by distributed object computing. Messaging, heterogeneity, failures, load balancing, and deadlocks are a few examples. Distributed object computing also has some new challenges: for example, reconciling different object models and languages, and object distribution and control.

DSOM and the emerging OMG standards help cope with many of the challenges. DSOM provides transparent messaging; the users simply make method calls without worrying about messages. In this respect, it it analogous to RPCs (remote procedure calls). DSOM provides location transparency which eases development of distributed applications. DSOM supports heterogeneity and multiple transports. Above all, DSOM offers a high degree of tailorability.

SOM/DSOM also solves some of the new problems introduced by object computing. SOM is a language-neutral object model and hence can be the common "lingua franca" of objects developed in various object-oriented languages. SOM scales well from fine-grained objects to large-grained objects. The emerging object services and common facilities standards from OMG address several other problems like object transactions, scalable directories, location transparent object creation via factories, etc. Future versions of SOMObjects are likely to incorporate these solutions.

PRO 0205

Introduction to REXX part 1

Chip Davis
Hilltop, Maryland

INTRODUCTION

Rexx is a wondrously powerful, yet intuitive and easy-to-use programming language. It may be an easy language to learn, but mastering any foreign tongue is easier if one is immersed in its usage. Syntax diagrams and lexical orthography are excellent tools for those interested in the structure and boundaries of a language, but they tend to overwhelm the novice or casual student. In this brief tutorial, we will demonstrate the fundamentals of Rexx by way of several small examples. This "conversational Rexx" approach will cover all the major components of the language and provide a foundation for further study.

Our primary vehicle will be an "Executive Decision Maker" program that will simulate, on your $3000 computer, the behavior of a one cent coin. That's just for starters, however. By the end of "Intro to Rexx - Part 2", we will have elaborated the program into a simulation of a $6.95 Magic 8-Ball.

There are many attractive things about the design of Rexx, and one of them is that Rexx code is inherently quite readable. (Just for giggles, refer to 7 and see if you can't pretty well figure out what's going on.) Of course, any code can be obfuscated, but with minimal attention to the naming of variables, your program can be practically self-documenting. Rexx uses English words for keywords and special characters for operators. It allows you to code in whatever mixture of case and spacing pleases your eye. Comments may go practically anywhere. There are no "reserved words"; if you happen to want to use a Rexx keyword as one of your variables, go right ahead. It will probably confuse you more than it will Rexx.

One of the most user-friendly aspects of Rexx is that you don't have to worry about data types. There's no need to convert a decimal number to its character equivalent just to print it out, nor to verify that two numeric values have the same internal representation before performing arithmetic or comparison operations on them. To Rexx, everything is simply a string of characters. Since the vast majority of data manipulations these days are string operations, this was a smart choice.

There are many other features of the design of Rexx that make it a remarkable language, such as arbitrary precision arithmetic, associative data storage, and its independence from the underlying hardware, character set, and operating system. These features are responsible for the power, flexibility, and universality of Rexx, but they don't make it hard to learn.

SOME BASICS

Look at the first line in Listing 1. This is a Rexx COMMENT. Everything within and including the '/*' and its companion '*/' is ignored. On most platforms, the operating system requires that a Rexx program start with a comment in column 1 of line 1. Other comments may begin at any point. Comments may span multiple lines and contain embedded comments. Comments may even be embedded in the middle of Rexx instructions, but this usually isn't such a hot idea.

The second and third lines are ASSIGNMENT instructions. The variable "heads" is given the value "1", and "tails" the value "2". In OS/2, variable names may be up to 250 characters long, and consist of the alphanumeric characters (upper and lower case) plus the characters '_', '?', '!', and '.'. They may not start with a digit or period, however. It doesn't matter what case you use for the variable name, Rexx translates it to uppercase and uses only that one name, regardless of how you write it. For example, "heads", "Heads", HeAdS", "hEADs", and "HEADS" all refer to the same variable. Use the case that you like, but consistency is what's important.

Note that in an assignment instruction, there may be as many spaces as you desire (including none) between the variable name, the '=', and the expression on the right. The EXPRESSION is a powerful concept that shows up in many Rexx statements. An expression can be quite complex, involving multiple variables, constants, operators, and functions. Evaluation of such an expression will yield some sort of string, however, and that string will be what is used in the statement.

```
<Listing 1>

/* rexx1.cmd: Flip-a-coin Executive Decision Maker */ /*  1 */
  heads = 1                                           /*  2 */
  tails = 2                                           /*  3 */
  Say "What is your YES/NO question?"                 /*  4 */
  Pull quest                                          /*  5 */
  answer = Random(1,2)                                /*  6 */
  If answer = heads Then Say "Yes."                   /*  7 */
  Else Say "No."                                      /*  8 */
  Return                                              /*  9 */
```

An interesting language design point is that Rexx, the language, places essentially no limits on anything. It does require that any limits imposed by a particular implementation of Rexx be in round numbers. Thus, OS/2 Rexx places very few limits on anything, but when it has to, it tends to limit it to 250 characters. We bit-heads find this novel, having long since memorized the powers of 2 well beyond the 32nd exponent. Real people are more grateful.

Look at line 4. This is a Rexx KEYWORD instruction. There are only 23 keyword instructions in Rexx, of which some are essentially synonyms, others are rarely used, and a few are necessary only if you are getting really fancy (arbitrary precision arithmetic or dynamic error handlers, for instance). The average Rexx programmer only uses about ten or twelve on a regular basis.

What do you suppose the SAY keyword instruction does? Exactly: Rexx evaluates the following expression and then writes the resultant string to the screen. In this case, the following expression consists of a single LITERAL string, so there's really nothing to evaluate. A literal is anything between a pair of matching quotation marks. Rexx uses the characters in the literal exactly as given: spacing, special characters, and all (but without the delimiting quotes). In OS/2, the maximum length of a single literal string is 250 characters, but you may concatenate any number of them together to create a single string of unlimited length (within your memory constraints). Each 'Say' instruction writes a new line to the screen, but since Rexx has no idea how wide your screen is, the line may wrap.

The PULL keyword instruction on line 5 waits for a line from the keyboard, and assigns it to the variable "quest". If there are multiple variables specified on the 'Pull' instruction, the first blank-delimited word of the input is assigned to the first variable, the second word to the second variable, and so forth. This continues until Rexx runs out of either input words or variables to assign them to. All variables specified in the 'Pull' instruction will be assigned a value. If there are fewer words in the input string than variables, the remaining variables will be assigned the NULL string. This is a perfectly valid string; it simply has a length of zero. You may explicitly use the null string by placing two quote characters immediately adjacent (""). If the 'Pull' instruction runs out of variables before it has assigned all the words of the input string, the last variable in the list is assigned all the remaining characters. (Rexx doesn't want you to accidentally lose anything.) In this program, we aren't interested in the words of the question being asked (frankly, we aren't even interested in the question at all), we just want the entire string to go into a single variable. Since "quest" is, by definition, the last variable in the list, it will be given all the remaining characters (which just happens to be all of them).

In line 6, we have another assignment instruction, but this time the expression on the right consists of a FUNCTION invocation. We (and Rexx) know this because we see something that looks for all the world like a variable, but it has a set of parentheses appended to it. Functions are wonderful little black boxes of programming: you feed them arguments and they return you a value. You don't have to know about the actual mechanics of its internal operation; it's magic. If this sounds suspiciously like the description of a SUBROUTINE, in truth there's not much difference between them. We'll discuss what differences there are in Part 2. For now, one of the differences is: where does the returned value end up? For a function, the returned value replaces the function invocation, and expression evaluation continues. This is handy, because it allows us to invoke multiple functions in the same expression, to nest functions within functions, and all sorts of other neat stuff. The Rexx expression evaluator sorts it all out for us and eventually presents us with a string containing the result of its labors.

What does the 'Random()' function do? It returns a random number, of course. The arguments we specified ("1" and "2") limit the range of the random number (in this case, to an integer between one and two, thus only two possibilities). The bigger question is: how do you find out what a particular function does, or what functions are out there for me to use? Okay, that's two questions, but the answer is the same: RTFM. On-line Help files now supply much of the content of the printed manuals, but these days my OS/2 desktop is as cluttered as my office desk ever was, so I like to be able to reach for a book and open it in my lap. Besides, you can't (yet) put Little Yellow Sticky Notes on a page of a Help file. OS/2 sports 76 built-in functions (77 if you count an alternative spelling of the 'Center()' function). In addition to the built-in functions, every Rexx implementation has one or more external function packs that are specific to a particular platform, software package, or environment. There are also some general-purpose function packages that supply oft-needed (but difficult to write yourself) routines. REXXLIB from Quercus Systems is a good example of a commercial, well-supported function library, and various free- and shareware libraries can be found around the Internet.

The IF keyword instruction on line 7 introduces us to our first compound Rexx instruction. There are two subkeywords (THEN and ELSE) which are keywords only within an 'If' instruction. The expression that follows the 'If' and precedes the 'Then' is evaluated and the result is presented to the 'If'. The 'If' is only looking for a "True" or "False" indication, so any other value is considered an error. This indication is represented by the values "1" for "True" and "0" for "False". Rexx doesn't care how you generate this "1" or "0" value in your expression, but the easiest way is to use a COMPARISON OPERATOR. A comparison operator simply compares the result of the expression on its left with the result of the expression on its right and returns a "0" or "1" as appropriate. In this particular example, since we don't use the variable "answer" anywhere else, we could have simply replaced "answer = heads" with "Random(1,2) = heads" in the 'If' expression. The program would have worked exactly the same. Comparison operators come in all the usual algebraic flavors: less-than (<), equal-to (=), greater-than (>), less-than-or-equal-to (<=), not-less-than (\<), and so forth. When the 'If' sees a "1", it executes the instruction following the 'Then' subkeyword, and skips the instruction following the 'Else' subkeyword (if any). When it sees a "0", it skips the 'Then' and its instruction, and executes the 'Else' instruction, if there is one. The 'Then' (with its instruction) is required, but an 'Else' (with its instruction) is not. If the 'If' expression is false, and no 'Else' is supplied, execution simply continues with the instruction following the 'Then' instruction.

The last line of our program also should be fairly self-evident. It causes control to RETURN to whomever invoked this program.

Now let's look at what this program does when it's invoked (Screen 1).

```
<Screen 1>

[D:\]rexx1
What is your YES/NO question?
Is this fun, or what?
Yes.
```

DO GROUPS

Okay, big deal. It did exactly what we expected it to do. This is Rexx. Get used to it.

```
<Listing 2>

/* rexx2.cmd: Flip-a-coin Executive Decision Maker */ /*  1 */
  heads = 1                                           /*  2 */
  tails = 2                                           /*  3 */
  Say "What is your YES/NO question?"                 /*  4 */
  Pull quest                                          /*  5 */
  If Right(quest,1) = '?' Then Do                     /*  6 */
    answer = Random(1,2)                              /*  7 */
    If answer = heads Then Say "Yes."                 /*  8 */
    Else Say "No."                                    /*  9 */
  End                                                 /* 10 */
  Else                                                /* 11 */
    Do                                                /* 12 */
    Say "That's not a question."                      /* 13 */
    Say 'A question ends in a "?"'                    /* 14 */
    End                                               /* 15 */
  Return                                              /* 16 */
```

Let's gussy it up a little. Look at Listing 2. Even though we don't care what it was, we want to make sure the user actually entered a question. We can determine this by invoking a built-in Rexx function that returns the specified number of right-most characters of a given string. In this case, all we are interested in is the last character, and it should be a question mark. If it is, we'll assume that whatever gibberish precedes it is a question, and provide an answer. Otherwise, we'll respond with a condescending error message. The problem is that our 'If' statement now has many more than the allowed "one statement following a 'Then' or 'Else' subkeyword". Sounds like a job for the DO-GROUP. By the simple expedient of surrounding any arbitrary number of statements with DO and END, those statements will be executed as a group, and satisfy the 'If' instruction rule.

By the way, if you really want to incite an Indentation Jihad, suggest to another Rexx programmer that the 'End' should align with the rest of the instructions in the group. Or not align. Or that the 'Do' and 'End' should both align with the group. Or whatever aberration you may prefer over "The One True Way To Indent a Do-End Group", which of course, is MY way of indenting. You'll find that the only one who doesn't have a firmly held conviction is Rexx, and Rexx couldn't care less.

Note the 'Say' instruction on line 14. Most languages require you to double-up a quotation mark if you want to put one inside a literal string delimited by them. Rexx will allow that, but provides a much better solution: use the other quotes. Rexx allows either single or double quotation marks as literal string delimiters. To imbed a double-quote inside a literal string, use single-quote characters to delimit the string, and vice versa.

We run this program and get Screen 2.

<Screen 2>

```
[D:\]rexx2
What is your YES/NO question?
The 'end' must line up under the 'do'!
That's not a question.
A question ends in a "?"

[D:\]rexx2
What is your YES/NO question?
Don't you agree?
```

SELECT

Listing 3 reflects that Life is never as simple as we'd like it to be. We need to simulate a three-sided coin, so we can have a third answer ("Maybe"). It's trivial to change the upper limit argument to the 'Random()' function on line 7, but this will make our 'If' instruction more complicated. The Rexx 'If' instruction stubbornly refuses to be swayed by the virtues of multi-valued logic; it insists on getting a "1" or "0" back from its expression. The traditional method of dealing with this situation was to nest 'If' instructions. This gives rise to code with "If ... Else If ... Else If ... Else ..." constructs, which gives rise (in almost direct proportion to the number of nests) to logic errors. In Rexx, it's simple: if you have only two choices, use an 'If'. If you have more than two choices, use the SELECT instruction.

<Listing 3>

```
/* rexx3.cmd: Flip-a-coin Executive Decision Maker */ /*  1 */
  heads = 1                                           /*  2 */
  tails = 2                                           /*  3 */
  Say "What is your question?"                        /*  4 */
  Pull quest                                          /*  5 */
  If Right(quest,1) = '?' Then Do                     /*  6 */
    answer = Random(1,3)                              /*  7 */
    Select                                            /*  8 */
      When answer = heads Then Say "Yes."             /*  9 */
      When answer = tails Then Say "No."              /* 10 */
      Otherwise Say "Maybe."                          /* 11 */
    End                                               /* 12 */
  End                                                 /* 13 */
  Else Do                                             /* 14 */
    Say "That's not a question."                      /* 15 */
    Say 'A question ends in a "?"'                    /* 16 */
    End                                               /* 17 */
  Return                                              /* 18 */
```

The 'Select' and its companion 'End' instruction delimit a list of WHEN instructions. Each 'When' instruction behaves exactly like an 'If' instruction, except that it doesn't have an 'Else' instruction. The expression following the 'When' is evaluated and the result is returned. If the result is a "0", the next 'When' in the list is executed. If it is a "1", the corresponding 'Then' instruction (or 'Do-End' group) is executed, and the 'Select' is terminated. Note this well: only the first successful 'When' instruction will have its 'Then' instruction executed; the rest will be skipped. If none of your 'When' instructions are successful, the 'Select' will give you an error. To avoid this, use an OTHERWISE instruction to cover any possibilities not addressed by your 'When' list. The 'Otherwise' should follow your last 'When' instruction. I like to have a 'When' instruction for everything I can think of, and then use an "Otherwise Say 'Call Chip! He screwed up something here!'" instruction. It's always a good idea to use an 'Otherwise' instruction, even if you don't have anything specific to do.

The 'Otherwise' keyword must be followed by an instruction (same as the 'Then' or 'Else'), so we need a "do nothing" instruction as a placeholder. This is the sole purpose of the NOP keyword instruction. 'Nop' does absolutely nothing; it makes IEFBR14 look productive. But it will stop a 'Select' from complaining if none of the 'When's were true (which may be a perfectly reasonable situation in your code).

Now let's run this a couple of times (see Screen 3).

```
<Screen 3>

[D:\]rexx3
What is your question?
Do you think they followed all that?
No.

[D:\]rexx3
What is your question?
Rats!
That's not a question.
A question ends in a "?"

[D:\]rexx3
What is your question?
Should I take a break?
Maybe.
```

INDEXED DO LOOPS

Well, that was fun. But it's kind of a pain to have to reinvoke our program from the command line each time we have a new question. Let's make it loop. The easiest way to do this is to enclose the major part of the program in a DO-LOOP (lines 7-21 of Listing 4). We don't want to repeatedly initialize our two variables, so we leave

```
<Listing 4>

/* rexx4.cmd: Flip-a-coin Executive Decision Maker */ /*  1 */
  heads = 1                                           /*  2 */
  tails = 2                                           /*  3 */
  Say "How many questions do you have?"               /*  4 */
  Pull numqs .                                        /*  5 */
  If \Datatype(numqs, "Whole") Then Return            /*  6 */
  Do num = 1 To numqs By 1                            /*  7 */
    Say ""                                            /*  8 */
    Say "What is question number" num"?"              /*  9 */
    Pull quest                                        /* 10 */
    If Right(quest,1) \= '?' Then Do                  /* 11 */
      Say "That is not a question."                   /* 12 */
      Return                                          /* 13 */
    End                                               /* 14 */
    answer = Random(1,3)                              /* 15 */
    Select                                            /* 16 */
      When answer = heads Then Say "Yes."             /* 17 */
      When answer = tails Then Say "No."              /* 18 */
      Otherwise Say "Maybe."                          /* 19 */
    End                                               /* 20 */
  End num                                             /* 21 */
  Say ""                                              /* 22 */
  Say "Goodbye."                                      /* 23 */
  Return                                              /* 24 */
```

them outside, along with a friendly "Goodbye" message to indicate we're through. This makes it look like one big 'Do-End' group, but the 'Do' instruction has some additional goodies that tell it how many times the instruction group should be executed. Since we don't want to limit the user to a fixed number of questions, the first thing we have to do is find out how many questions the user has. Notice that the list of variables following the 'Pull' keyword on line 5 now has a "." in it. This is a placeholder that says "Ignore the data that would have gone into a variable at this position." Think of it as the bit bucket. You can have as many of these placeholders as you want, so you don't have to waste storage with a bunch of "junk1", "junk2", "junk3", ... variables. In this case, we are only interested in the first blank-delimited "word". (Actually, we hope it's a number, and a non-negative integer at that.) After all, there's nothing to prevent a (l)user from responding to our prompt with "5 or 6, maybe as many as 10", is there? So we snatch off the first word, and throw the rest of the characters away. (Remember the special rule about the last variable in the list?) There's still no guarantee that our user didn't respond with "five", or something else that's clearly not an integer. The 'Do' instruction really doesn't have any idea how to go about looping 3.56 times, or -23 times for that matter. (It's been suggested that Rexx could execute the instructions in the group in reverse order 23 times, but I don't know...)

How do we tell if a variable contains what it claims to hold? There's a handy little built-in function named 'Datatype()' that will tell us. Pass 'Datatype()' the value in question, and one of the keywords that indicates what you hope it is, and the function will return you a "1" (True) if it is, and a "0" (False) if it ain't. In this case, we want to see if "numqs" is a "Whole" number (AKA "integer"). Our only problem now is that we want to do something ('Return') if it isn't, so we have to set up our 'If' test appropriately. One way to do this would be to code 'Then Nop' with 'Else Return', but that's kind of kludgey.

Just as numbers may have a PREFIX OPERATOR such a "+" or "-", a logical value may have the NOT prefix operator "\" which inverts the value: a True becomes False, and vice versa. Notice that there's no need to have a comparison operator in this 'If' expression, since the 'Datatype()' function only returns a "0" or "1" (which is then inverted by the "\" prefix operator). What other types of data can you check with the 'Datatype()' function? For starters, you can verify that it consists of only alphanumeric, only uppercase, only lowercase, or only mixedcase characters, is a valid Rexx decimal, hexadecimal, or binary number, and more. If you want to know more, you know the drill ...

At this point in our program, we have a valid integer (although it may be zero or negative). The simplest form of Do-loop would be something like "Do 5", which tells Rexx to execute the group of instructions between the 'Do' and 'End' five times, then continue with the rest of the program. In our case, "Do numqs" would loop as many times as the number in the variable indicates. What's really happening in the 'Do' instruction is that Rexx evaluates the expression after the 'Do' keyword, and if it ends up being a non-negative integer, that's how many times it will loop. The 'Do' expression could be a very complicated calculation to determine the loop count. In addition to looping through the instructions "numqs" times, we want to keep a counter so that we can number each question. This can be done easily with another variable, but we would have to initialize it, and increment it each time. The Do-loop will do this for us if we ask. The "num = 1" on line 7 is a form of assignment instruction. It instructs Rexx to initialize the variable num to the value of the expression on the right, which just happens to be "1". Having set up a loop index variable, we can now use it to decide when to terminate the loop. Rexx evaluates the expression following the TO subkeyword and uses that as the upper bound of the value of "num". In other words, when the "num" value exceeds the 'To' expression value, the loop will stop. That sort of presumes that something is incrementing "num" each time through the loop, and that is exactly what happens. By default, the index variable is incremented by "1" after each pass through the loop.

You may, of course, increment (or decrement) the index variable by any amount you wish. Rexx evaluates the expression following the BY subkeyword, and adds that value to the index variable.

All of these expressions are evaluated just prior to beginning the loop. If you were to make any change to the variables involved (generally a bad idea, anyway), it will have no effect on the operation of the loop. A major advantage that the loop variable has over "Do numqs" is that a loop variable is not limited to non-negative integers. It may contain any numeric value you choose: positive, negative, fractional, scientific notation, anything as long as it's base 10.

Notice that on line 21, the index variable is specified on the ‘End’ instruction. This helps us (and Rexx) keep track of things when we have multiple nested loops.

Back up on line 8, we ‘Say’ a null line. This simply leaves a blank line before the next ‘Say’ line. In line 9, we have changed the prompt slightly to indicate the question number. Our ‘Say’ finally has an expression to evaluate. I said at the outset that Rexx uses special characters for its operators, and that's true. One of the more radical notions in Rexx, however, is the string CONCATENATION operator. It's the blank character. But get a load of this: in some cases, it's absolutely nothing!

As was mentioned earlier, Rexx was designed to make string manipulations easy. To that end, the language tries to make string operations as natural as possible. Let's take two completely contradictory concepts (“Basic” and “Advanced”) and juxtapose them to create one of my favorite oxymorons: Advanced Basic. In English, do we say “Advanced” | | ” “ | | ”Basic”, or “Join(“Advanced”,”Basic”,*)”, or (heaven help us) “&CONCAT OF &ADVANCED &BLANK &BASIC”? Of course not. We simply write the two terms separated by a blank. In Rexx, this is called BLANK CONCATENATION and what could be more natural? If you want the other kind of concatenation, the “jam'em up together” flavor, Rexx gives you two ways to do that. The simplest way is to just write the two terms side-by-side. Since human languages don't have to make a distinction between constants and variables, Rexx does require that we delimit literals with quotes. Anything that isn't enclosed in quotes, and is a legal variable name, will have its value substituted for it. Essentially, if Rexx can tell what's a literal and what's a variable, you may perform ABUTTAL concatenation by simply writing one term adjacent to the other. We've used both forms of concatenation in line 9. The string “What is question number” will be concatenated to the value of the variable “num”, with a blank between them. The literal string “?” will then be concatenated without a blank to the resultant string. There will be times when you want to abut the values of two variables. Obviously, if you use abuttal, Rexx will think you are referring to another variable altogether. For this situation, Rexx allows the double-vertical-bar (| |) concatenation operator. It can be used for all concatenation operations, at the expense of readability.

Let's see how this works (Screen 4).

```
<Screen 4>

[D:\]rexx4
How many questions do you have?
5

What is question number 1?
Are you thoroughly Warped?
Yes.

What is question number 2?
Will Windows95 actually ship this year?
Maybe.

What is question number 3?
You gotta be kidding!
That is not a question.

Goodbye.
```

ITERATE

Unfortunately, we have a minor, er, feature. If what you entered wasn't a question, it kicks you out. It would be much nicer if it simply gave you the message and went back through the loop again.

Perhaps you've noticed that you haven't seen any "goto" instructions in Rexx. There aren't any. "goto"s are the bane of programming, leading to spaghetti code, logic errors, geek-speak, Snickers-and-Diet-Coke cravings, and other aberrations. Rexx will have none of it. "goto"s wreak havoc with the control structures of a Do-loop, so Rexx has specific mechanisms for changing the flow of execution within a loop. The ITERATE keyword instruction tells Rexx to discontinue executing the loop instructions and continue from the top of the loop, where index variable incrementing and checking are performed as usual. If you have nested loops, the loop that is iterated is the one the 'Iterate' instruction is immediately inside. You may, however, iterate an outer loop by specifying that loop's index variable on the 'Iterate' instruction. This will terminate all loops internal to the one iterated. Obviously, only Do-loops can be iterated; you can't 'Iterate' a Do-group.

```
<Listing 5>

/* rexx5.cmd: Flip-a-coin Executive Decision Maker */ /*  1 */
  heads = 1                                           /*  2 */
  tails = 2                                           /*  3 */
  Say "How many questions do you have?"               /*  4 */
  Pull numqs .                                        /*  5 */
  If \Datatype(numqs, "Whole") Then Return            /*  6 */
  Do num = 1 To numqs By 1                            /*  7 */
    Say ""                                            /*  8 */
    Say "What is question number" num"?"              /*  9 */
    Pull quest                                        /* 10 */
    If Right(quest,1) \= '?' Then Do                  /* 11 */
      Say "That is not a question."                   /* 12 */
      Iterate num                                     /* 13 */
    End                                               /* 14 */
    answer = Random(1,3)                              /* 15 */
    Select                                            /* 16 */
      When answer = heads Then Say "Yes."             /* 17 */
      When answer = tails Then Say "No."              /* 18 */
      Otherwise Say "Maybe."                          /* 19 */
    End                                               /* 20 */
  End num                                             /* 21 */
  Say ""                                              /* 22 */
  Say "Goodbye."                                      /* 23 */
  Return                                              /* 24 */
```

Let's see if that improves matters (line 13 of Listing 5, and Screen 5).

```
<Screen 5>

[D:\]rexx5
How many questions do you have?
4

What is question number 1?
Would you like a Pan-Galactic Gargle Blaster?
Yes.

What is question number 2?
Too bad, I'm fresh out.
That is not a question.

What is question number 3?
So it wasn't. Is this?
Maybe.

What is question number 4?
Oh well.
That is not a question.

Goodbye.
```

CONDITIONAL DO LOOPS

Much better, but it's somewhat unrealistic to expect that users consulting oracular software would know in advance how many questions they will want to ask, now isn't it? You could make a better case that oracular software, if it's any good, should already know how many questions will be asked. A compromise is to answer questions until some sort of terminating message is received. We'll modify our program to loop until the question is the word "quit". Let's see, how should we write that in Rexx? How about: "Do Until quest = 'QUIT'", like on line 5 of Listing 6? The major difference between the UNTIL form of loop termination and the 'To' form, is that the expression in the 'Until' is evaluated each time around the loop. This means that changes made to a variable used in the 'Until' expression could affect the iteration of the loop.

```
<Listing 6>

/* rexx6.cmd: Flip-a-coin Executive Decision Maker */ /*  1 */
  heads = 1                                          /*  2 */
  tails = 2                                          /*  3 */
  Say "To exit, enter ‘quit’."                       /*  4 */
  Do num = 1 Until quest = ‘QUIT’                    /*  5 */
    Say ""                                           /*  6 */
    Say "What is question number" num"?"             /*  7 */
    Pull quest                                       /*  8 */
    If Right(quest,1) \= ‘?’ Then Do                 /*  9 */
      Say "That is not a question."                  /* 10 */
      Iterate num                                    /* 11 */
    End                                              /* 12 */
    answer = Random(1,3)                             /* 13 */
    Select                                           /* 14 */
      When answer = heads Then Say "Yes."            /* 15 */
      When answer = tails Then Say "No."             /* 16 */
      Otherwise Say "Maybe."                         /* 17 */
    End                                              /* 18 */
  End num                                            /* 19 */
  Say ""                                             /* 20 */
  Say "Goodbye."                                     /* 21 */
  Return                                             /* 22 */
```

Are you bothered by the fact that we haven't given the variable "quest" a value before comparing it to a literal string? There are two answers to that. One answer is that "quest" actually does have a value. All variables in Rexx have an initial, default value. It is simply the variable name in uppercase. Many Rexx programmers rely on this to define constants for them, but it's considered poor form. If the "quest = 'QUIT'" test were performed on entry to the loop, there would be a valid value ("QUEST") in the variable, and the expression would not fail. However, the other answer is that the 'Until' test isn't performed until the instructions in the loop have been executed once. Therefore, the value of "quest" on entry to the loop is irrelevant, because the 'Until' condition is not tested until the bottom of the loop, effectively. It should be obvious from the word "until" that as long as the expression returns a "False", the loop will continue.

There is a complementary form of dynamic termination testing that uses the WHILE keyword. As you might expect, as long as its expression returns a "True", the loop will continue. The other difference is that the 'While' condition is indeed tested before the loop is started, at the top of the loop, effectively. Thus, if the 'While' expression is "True", the instructions in the loop are skipped altogether. You can't use them both in the same Do-loop, but you can use either one of them with an index variable, as we've done here.

Don't forget to tell the user what to say to get out of this otherwise endless loop (line 4).

You may be wondering why we are testing for the string “QUIT” in all caps. This is an artifact of the ‘Pull’ instruction in that it uppercases the input string before assigning the data to variables. This is handy here, because we are looking for a particular string of characters. We don’t want to have to test for every permutation of “quit”, “Quit”, “qUiT”, ad nauseum.

Let’s see if this works (Screen 6).

```
<Screen 6>

[D:\]rexx6
To exit, enter ‘quit’.

What is question number 1?
Will this really stop when I say ‘quit’?
Maybe.

What is question number 2?
Okay then, quit.
That is not a question.

What is question number 3?
Does it have to be the only thing on the line?
Yes.

What is question number 4?
QuiT
That is not a question.

Goodbye.
```

LEAVE

Well, at least it quit. A cursory review of the logic in our loop reveals a minor flaw. Where do we check to see if “quest” is a valid question? Right after we pull it in. Where do we check to see if “quest” is “QUIT”? At the bottom of the loop, after we’ve put out the message that it’s not a question. What does this mean? It means that not everything that can be done on a ‘Do’ instruction is best done there. In this case, we are better off having an explicit check for our termination condition immediately after getting the string. Okay, fine. So how do we get out of the loop then, if Rexx has no “goto”? We use the Do-loop control keyword instruction LEAVE (line 9 of Listing 7). ‘Leave’ modifies the flow of control of its loop to effect a graceful termination, and to continue with the instruction following the ‘End’. Like ‘Iterate’, you may choose to terminate an outer-level Do-loop by specifying that loop’s index variable on the ‘Leave’.

<Listing 7>

```
/* rexx7.cmd: Flip-a-coin Executive Decision Maker */ /*  1 */
  heads = 1                                           /*  2 */
  tails = 2                                           /*  3 */
  Say “To exit, enter ‘quit’.”                        /*  4 */
  Do num = 1                                          /*  5 */
    Say “”                                            /*  6 */
    Say “What is question number” num”?”              /*  7 */
    Pull quest                                        /*  8 */
    If quest = ‘QUIT’ Then Leave num                  /*  9 */
    If Right(quest,1) \= ‘?’ Then Do                  /* 10 */
      Say “That is not a question.”                   /* 11 */
      Iterate num                                     /* 12 */
    End                                               /* 13 */
    answer = Random(1,3)                              /* 14 */
    Select                                            /* 15 */
      When answer = heads Then Say “Yes.”             /* 16 */
      When answer = tails Then Say “No.”              /* 17 */
      Otherwise Say “Maybe.”                          /* 18 */
    End                                               /* 19 */
  End num                                             /* 20 */
  Say “”                                              /* 21 */
  Say “Goodbye.”                                      /* 22 */
  Return                                              /* 23 */
```

If this works (Screen 7), we have a mildly amusing little toy to play with, and you have a fairly substantial understanding of the basics of Rexx.

<Screen 7>

```
[D:\]rexx7
To exit, enter ‘quit’.

What is question number 1?
Is that all there is?
No.

What is question number 2?
Do I have a question number 3?
Yes.

What is question number 3?
Sorry, I’m outta here.
That is not a question.

What is question number 4?
quit

Goodbye.
```

CONCLUSION

Stay tuned for "Intro to Rexx - Part 2". There we'll show you how to write your own subroutines and functions, use compound variables, do file I/O, issue commands to OS/2 and retrieve the responses, and do a little parsing.

PRO 0207

C++ Application Frameworks Under the Microscope

Larry O'Brien
Miller Freeman, Inc.
San Francisco, Calif.

Application Frameworks Under the Microscope

Understanding, Comparing, and Choosing
C++ Application Frameworks

Larry O'Brien
Editorial Director
Software Development magazine

Comparing Application Frameworks

- What is an application framework?
- What do they have in common?
- How can we distinguish between them?
- How do you choose which one to invest your time into learning?

What is an application framework?

- Application: a program that provides the complete set of functions necessary to accomplish a task
- What is a framework?
  - » A basic conceptual structure (as of ideas)
  - » A skeletal, openwork, or structural frame
  - » Frame of reference
  - » The larger branches of a tree that determines its shape

Typical Frameworks

- Borland's ObjectWindows Library
- IBM's Visual Age C++ Class Libraries
- Inmark Technology's zAPP
- Microsoft's Microsoft Foundation Classes
- XVT's Power++
- Zinc's zApp

Über-Frameworks

- Galaxy
  - » Virtual operating system allows OO abstraction of virtually all system-level functions
  - » Requires strategic commitment
- Taligent
  - » Secret, late, and big. Be skeptical.

The Framework Framework

- Control and Interface Classes
- Helper Classes
- Gurus
- Derived C&I Classes
- Domain Classes

Control & Interface Classes

- All use derivative of Smalltalk-80's Model-View-Controller control structure
- Model
  - » The document: What is stored to disk
  - » Other runtime objects needed to model the domain
- View: Visual display
- Controller: Generally not modeled. Keyboard, Mouse, Pen, etc.

Helper Classes

- Storage Mechanism
- String
- Collection classes
- Time, Date, Money
- Invoices? Telephone Calls?

Gurus

- Application gurus
  - » Provide skeleton working app
  - » One-way
  - » Limited value
- Control gurus
  - » Link up OS events with C++ class calls
  - » Two-way
  - » High value

SOFTWARE Development

Derived Control and Interface Classes

- MyApplication, MyDocument, MyView
- Classes corresponding to dialog boxes, perhaps menus, status bars, toolbars, etc.
- In simple programs, majority of programming effort expended on these, *therefore*, Class Gurus good differentiator

Domain Classes

- User classes should be linked to auto-derived Document class via composition relationship
- Consider widespread use of Proxy pattern
  ("Design Patterns," Gamma, et al., A-W, 1995)
- In large programs, more significant than display issues and *therefore* Helper Classes good differentiator

Distinguishing Between Frameworks

- Control and Interface Classes
- Helper Classes
- Gurus
- Operating System Support
- Development Environment

What Questions Not To Bother Asking

- Will its revolutionary architecture reduce my programming time by 90%?
- Is it as easy to use as Visual Basic?
- Is it the really, truly object-oriented?

Looking at C & I Classes

- Most important for smaller projects
- Form-based development
- Front-end database development
- Look for implementation of O/S widgets (tabbed notebooks, etc.)
- Geometry management
- Look at standard, complex objects

For Instance, Fonts

- Font class good revelation of O/S assumptions

SOFTWARE Development

Looking at Helper Classes

- Most important for larger projects
- Complex data structures
- System-level stuff
- Look for consistent implementation in collection classes
- Look at how well they implement the basics

For instance, Strings

- String class helps indicate relative effort on helper classes

Looking at Gurus

- "Like a cocktail-lounge mentalist, a software wizard asks leading questions, then jumps to conclusions." -- Larry Constantine, SD mag, September 1994
- "For the initial learning-and-using encounter period (a well to which we return to guzzle from as we desire to increase our knowledge), [gurus] are available to frame usage within the context of predefined tasks." -- Gen Kiyooka, SD mag, January 1995

The Nice Things About Gurus

- Perform simple operations flawlessly
- Are not fazed by multi-file relationships
- The simple chores are often where we're most careless
- They write consistent comments

The Bad Things About Gurus

- App Gurus are one-way
- Not flexible
- Don't expose internals
- Concentrate on auto-deriving C&I classes and attaching messages to functions
  - » (which isn't hard)

I Want Domain Level Gurus!

- Choosing right class for data structure
- Recoding predefined functions
- Performing common tasks

Looking at O/S Support

- Forget promises. What do they support now?
- Are development tools hosted on all platforms? If not, why not?
- Delays between releases on different platforms?

Analyzing Frameworks

- Design review
- Code Metrics
  - » Cyclomatic Complexity
  - » Volatility
  - » Software Science Length
  - » OO Code Metrics
    - – Depth of inheritance
    - – Coupling
    - – Cohesion

Looking at Development Environments

- Access to gurus
- Interface builders primarily for form-based development
  - » Is this really worth all the attention it gets?
- Run-time exposure of objects
- Integration with other tools
  - » Obviously, vendor-supplied has huge advantage here

Design Review

- "Framework framework" holds up very well.
- Exception: Utah and, probably, Taligent, where control structure seems significantly different than MVC-derivative

"Classic" Metrics

(Cyclomatic Complexity, Software Science)

- Very significant differences between frameworks.
- Good for identifying volatility, vendor emphasis
- They conclude these things aren't that complex: ~200-1000 hours to duplicate
  - » This is all pretty "stupid" code
  - » However, OO relationships mask significant (maybe majority) complexity

"OO" Metrics

- Intriguing work
- No tools available
- If only vendors would open up their parse information!

SOFTWARE Development

Application Frameworks Under the Microscope

Understanding, Comparing, and Choosing C++ Application Frameworks

SOFTWARE Development

this page intentionally left blank

SYS 0201

OS/2 for the Power PC, the IBM Microkernel Architecture: The Big Picture

Miles Barel
IBM
Boca Raton, Flor.

OS/2 WARP

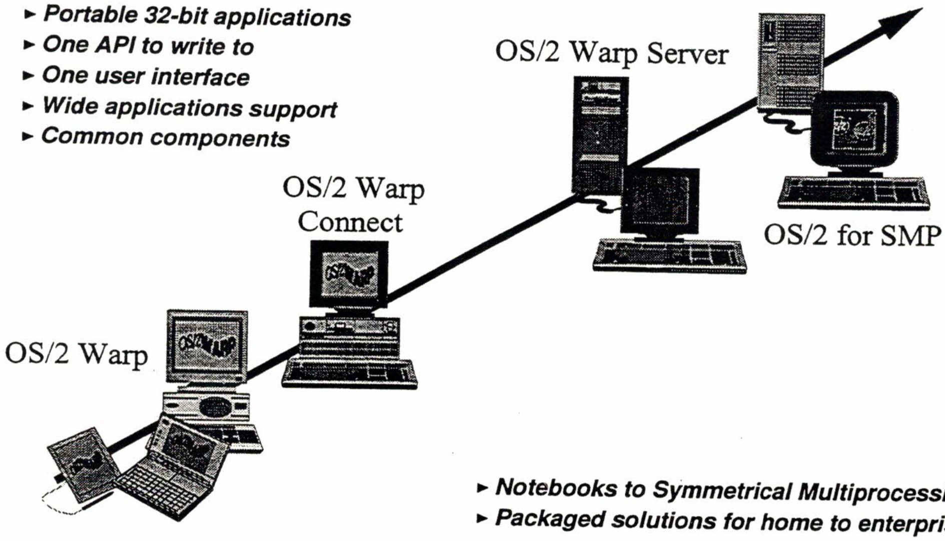
IBM's Market Leading 32-bit Operating System

- A Full Productivity Environment - Not Just an Operating System
- 32-bit = Reduce Cost and Increase Performance
- Full Support for Legacy DOS and Windows Applications
- Supports Modern Development Technologies
- Full Interoperability in Heterogeneous Networks
- A Complete Family of Products for All Needs

OS/2 WARP
OS/2 WARP Connect
OS/2 WARP Server

OS/2 WARP The OS/2 Warp Family

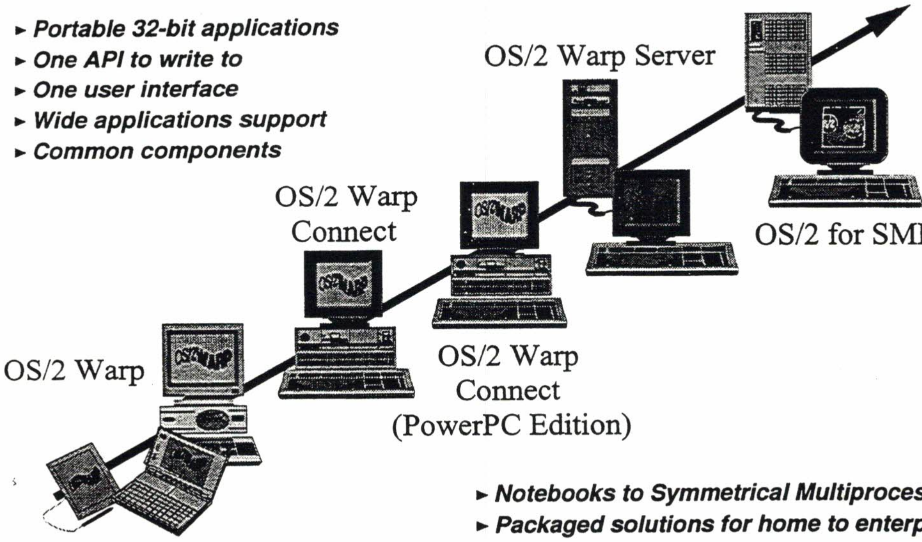
- *Portable 32-bit applications*
- *One API to write to*
- *One user interface*
- *Wide applications support*
- *Common components*


OS/2 Warp Server
OS/2 Warp Connect
OS/2 for SMP
OS/2 Warp

- *Notebooks to Symmetrical Multiprocessing*
- *Packaged solutions for home to enterprise*

OS/2 WARP The OS/2 Warp Family Explands to the PowerPC Platform

- *Portable 32-bit applications*
- *One API to write to*
- *One user interface*
- *Wide applications support*
- *Common components*


OS/2 Warp Server
OS/2 Warp Connect
OS/2 for SMP
OS/2 Warp
OS/2 Warp Connect (PowerPC Edition)

- *Notebooks to Symmetrical Multiprocessing*
- *Packaged solutions for home to enterprise*

OS/2 WARP OS/2 Warp Connect (PowerPC Edition)

Everything you've come to expect from OS/2 Warp:

- ***OS/2 Warp - Now Based on the IBM Microkernel***
  - ***The Same Look and Feel - Workplace Shell***
  - ***Preserves Heritage, Running OS/2, Windows 3.x & DOS Applications***
- ***Release 1 - Client Design Point:***
  - ***Reasonable RAM & DASD***
  - ***Broad Range of Client Support, Transports, DB2, LAN Server, ...***

Plus....

OS/2 WARP OS/2 Warp Connect (PowerPC Edition)

The benefits of the new PowerPC-based systems

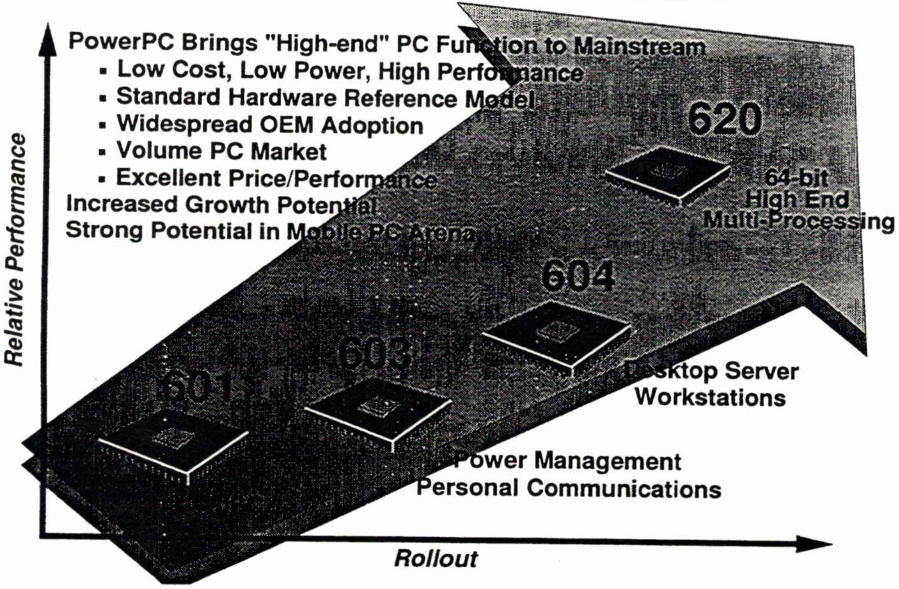
- Power to handle a new range of applications
- A platform designed for HUMAN-CENTERED applications
- The Price/Performance of RISC

An architecture designed for the demands of the future

- Scalable from embedded systems to clients to servers
- Modular, providing for flexible configurations and incorporation of new capabilities
- Designed for client / server solutions from the ground up
- Portable to future platforms

...Creating New Possibilities

OS/2 WARP PowerPC Microprocessor Family



PowerPC Brings "High-end" PC Function to Mainstream
- Low Cost, Low Power, High Performance
- Standard Hardware Reference Model
- Widespread OEM Adoption
- Volume PC Market
- Excellent Price/Performance

Increased Growth Potential
Strong Potential in Mobile PC Arena

620
64-bit
High End
Multi-Processing
604
Desktop Server
Workstations
603
601
Power Management
Personal Communications
Relative Performance
Rollout

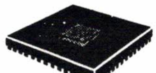
OS/2 WARP What Is Human-Centered?

- Technology centered around our natural senses and responses
- New paradigm of user-to-computer interaction
  - ✓ Processing of information from a variety of senses including sight, sound and touch
  - ✓ "Human Oriented Computing" rather than "Computing Oriented Humans"
- Allow users to work freely, intuitively, conversationally and cooperatively
  - ✓ Personal assistance instead of tool
  - ✓ Better focus on work tasks rather than how to use the machine
- Enables computing for a broader range of people

OS/2 WARP OS/2 Warp and the PowerPC Microprocessor

The Human-Centered Combination

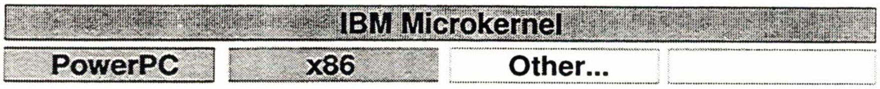
OS/2 WARP +



POWERPC

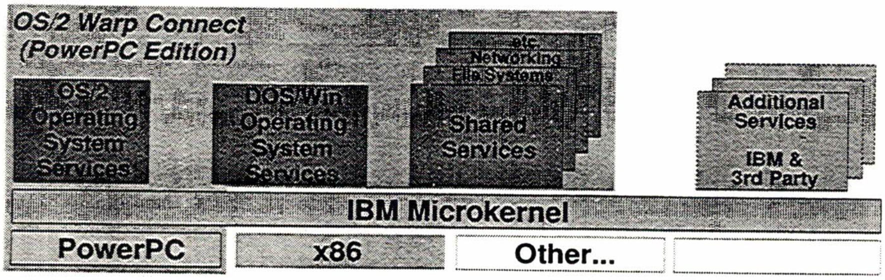
- Makes Human-Centered Technologies Accessible to All Users
  - ✓ Affordable
  - ✓ No Special Hardware Required
- Human-Centered Computing
  - ✓ Begins with Enablement
  - ✓ Continues with Exploitation
- Strong Potential In Mobile PC Arena

OS/2 WARP The IBM Microkernel


IBM Microkernel
PowerPC
x86
Other...

- Provides Fundamental O/S Functions
  - ✓ Memory Management
  - ✓ I/O
  - ✓ Process / Thread Management
  - ✓ ....
- Portable to a wide range of processors
- Scalable to diverse architectures
- Open to industry
  - ✓ Other H/W Platforms, Other Operating Environments

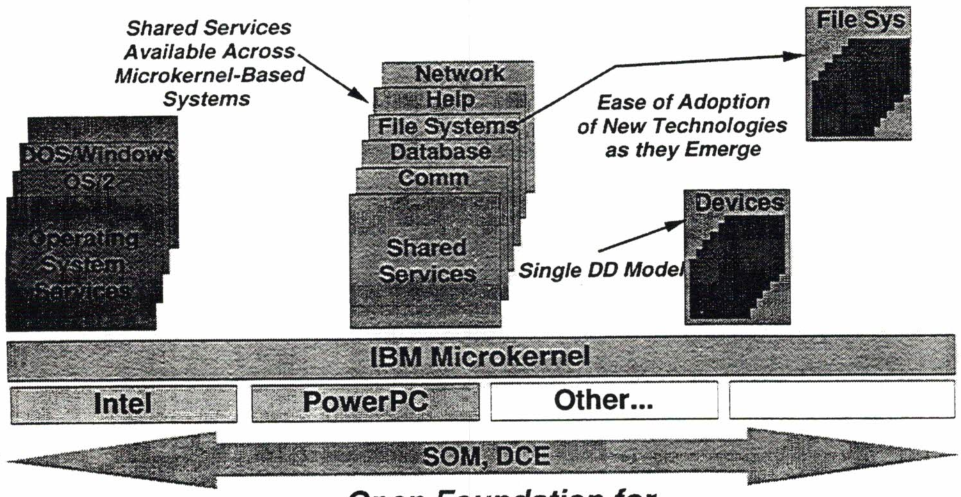
OS/2 WARP OS/2 Warp on the IBM Microkernel


OS/2 Warp Connect
(PowerPC Edition)
OS/2 Operating System Services
DOS/Win Operating System Services
etc
Networking
File Systems
Shared Services
Additional Services
IBM & 3rd Party
IBM Microkernel
PowerPC
x86
Other...

- Modular & Flexible
  - ✓ Based on Common System Elements
  - ✓ Allows Quick Response to New Technologies and Markets
- Specific Offerings for Different Markets
  - ✓ Taking Advantage of Commonality
- Modern OS Base, While Taking Existing Investment Forward

Designed for the future...

OS/2 WARP IBM Microkernel-Based Architecture Design Directions



Shared Services Available Across Microkernel-Based Systems

DOS/Windows
OS/2
Operating System Services

Network
Help
File Systems
Database
Comm
Shared Services

Ease of Adoption of New Technologies as they Emerge

File Sys

Single DD Model

Devices

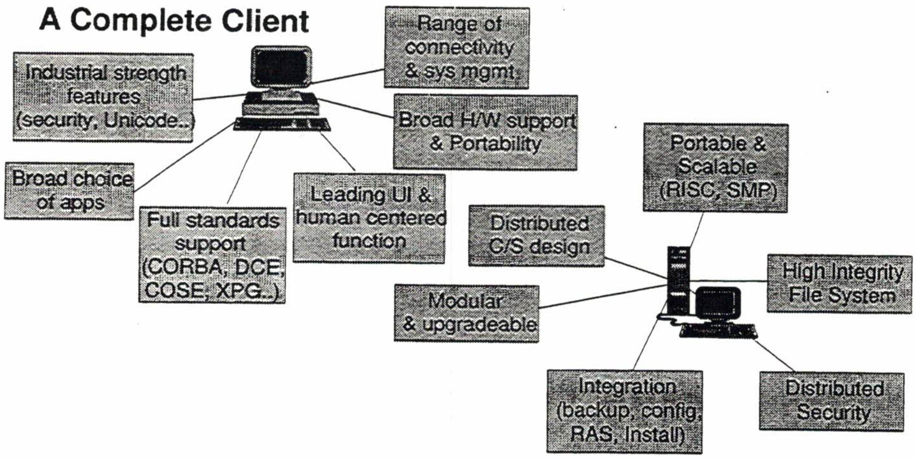
IBM Microkernel

Intel
PowerPC
Other...

SOM, DCE

Open Foundation for Industry Innovation and Investment Protection

OS/2 WARP Going Further on Client and Server



A Complete Client

Industrial strength features (security, Unicode..)
Range of connectivity & sys mgmt
Broad H/W support & Portability
Broad choice of apps
Full standards support (CORBA, DCE, COSE, XPG..)
Leading UI & human centered function

Portable & Scalable (RISC, SMP)
Distributed C/S design
High Integrity File System
Modular & upgradeable
Integration (backup, config, RAS, Install)
Distributed Security

A Scalable Server

OS/2 Warp Connect (PowerPC Edition) And the OS/2 User

Performance

- Combines the power of the PowerPC microprocessor with award winning OS/2 Warp

Compatibility

- Protects customer investments in DOS and Windows applications.
- OS/2 applications require a simple recompile to exploit the PowerPC microprocessor.

Value

- Provides a complete network client for LAN Server, NetWare (when available) and TCP/IP as well as productivity applications.

Ease of Use

- Enhances ease of use through an improved, more visually appealing interface, Human-Centered technologies, streamlined installation, and a total packaged connectivity solution.

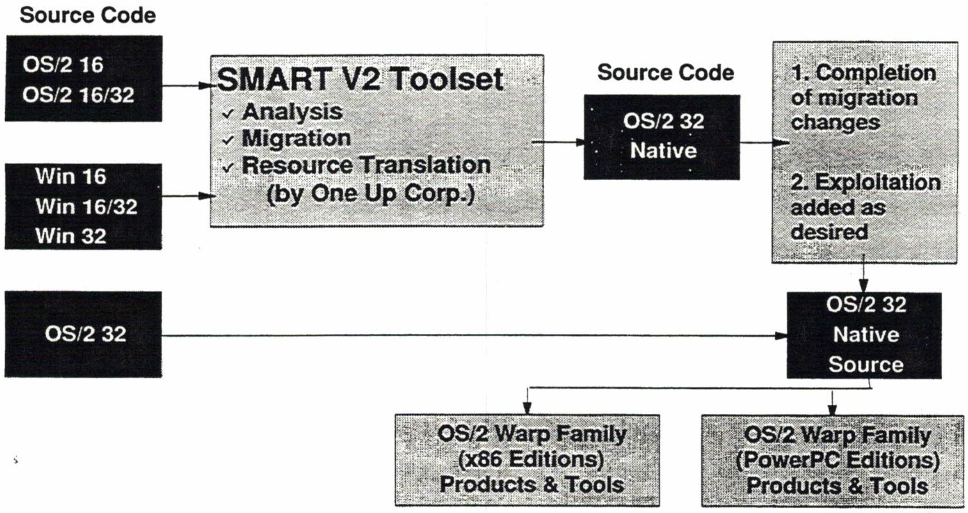
Reliability

- Builds upon the proven, stable, reliable OS/2 code base to create a networking package that business customers can count on.

OS/2 Warp Connect (PowerPC Edition) And the Application Developer

- **New Applications / New Opportunities**
  - ✓ **Performance characteristics of the PowerPC microprocessor enables the development of more exciting, more robust applications**
    - **Add Human-Centered Features**
    - **New Application Opportunities (e.g., Collaborative Computing)**
  - ✓ **Further Exploitation of Reuse**
  - ✓ **New Markets for Existing Applications**
- **OS/2 32-bit API**
  - ✓ **Portable: x86 <--> PowerPC**
  - ✓ **Single Source Between Platforms**
  - ✓ **16 --> 32-bit Migration Tools**

OS/2 WARP Application Migration Strategy
S.M.A.R.T.



Source Code

OS/2 16
OS/2 16/32

Win 16
Win 16/32
Win 32

OS/2 32

SMART V2 Toolset
- ✓ Analysis
- ✓ Migration
- ✓ Resource Translation (by One Up Corp.)

Source Code

OS/2 32 Native

1. Completion of migration changes

2. Exploitation added as desired

OS/2 32 Native Source

OS/2 Warp Family (x86 Editions) Products & Tools

OS/2 Warp Family (PowerPC Editions) Products & Tools

OS/2 WARP The Power of the Future...*NOW*!

- Better Client & Server Operating Systems
- Wider Selection of Applications to Chose From
- More Complete Client/Server Enabling Platforms
- IBM Expertise in the Enterprise Environment
- Commitment to Leadership in Open Standards Support
  - For Better Integration in Multi-Vendor Environments
  - For Confidence You'll Have Freedom of Choice
    Now ... and in the Future...

"My personal favorite--perhaps the most useful productivity software available today--is IBM's superior operating system, OS/2."
Stratford Sherman
"PC Products That Can Change Your Life"
Fortune, March 6, 1995

"...you can have a high degree of scalability. We have the same installation in a 500-user network as in a two-user LAN"
Jose Manuel Aguirre
Banco Bilbao Vizcaya, Spain

OS/2® WARP

IBM

There Is A Difference

LAN 0305

Migrating from Netware to LAN Server: Why and How

No paper available. Please contact speaker

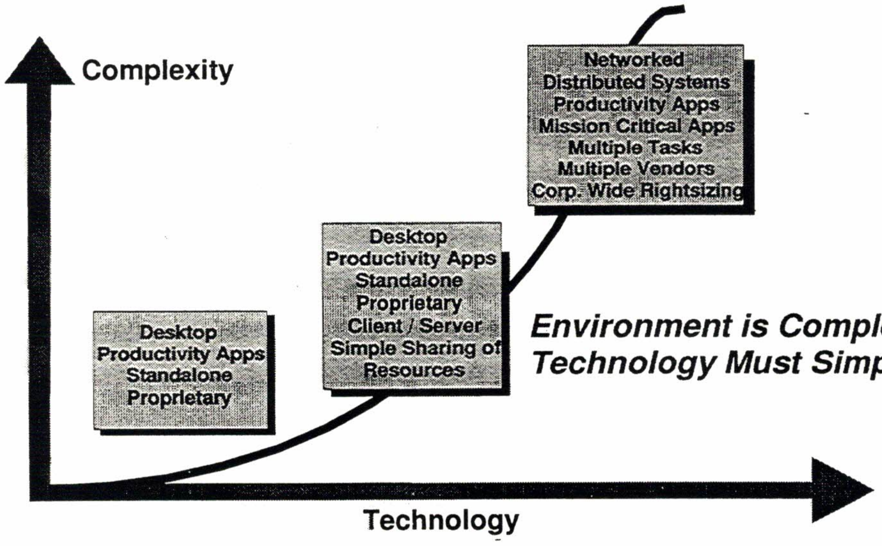
this page intentionally left blank

MGT 0303

OS/2 Strategy and Directions

Miles Barel
IBM
Boca Raton, Flor.

OS/2 WARP Personal Systems and Applications are Evolving..



Complexity

Desktop
Productivity Apps
Standalone
Proprietary

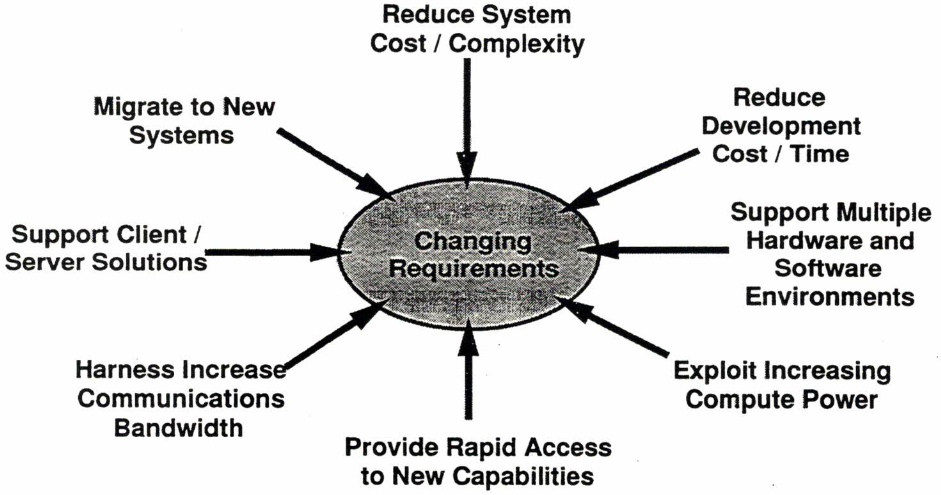
Desktop
Productivity Apps
Standalone
Proprietary
Client / Server
Simple Sharing of
Resources

Networked
Distributed Systems
Productivity Apps
Mission Critical Apps
Multiple Tasks
Multiple Vendors
Corp. Wide Rightsizing

Environment is Complex:
Technology Must Simplify

Technology

OS/2 WARP Future Imperatives



Reduce System
Cost / Complexity

Migrate to New
Systems

Reduce
Development
Cost / Time

Support Client /
Server Solutions

Changing
Requirements

Support Multiple
Hardware and
Software
Environments

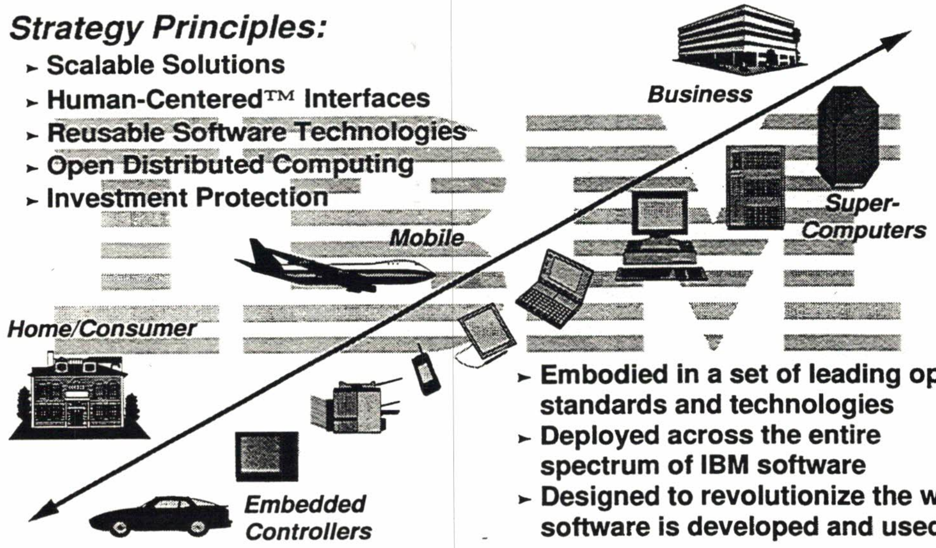
Harness Increase
Communications
Bandwidth

Exploit Increasing
Compute Power

Provide Rapid Access
to New Capabilities

OS/2 WARP IBM's Software Strategy A Customer Driven Strategy...

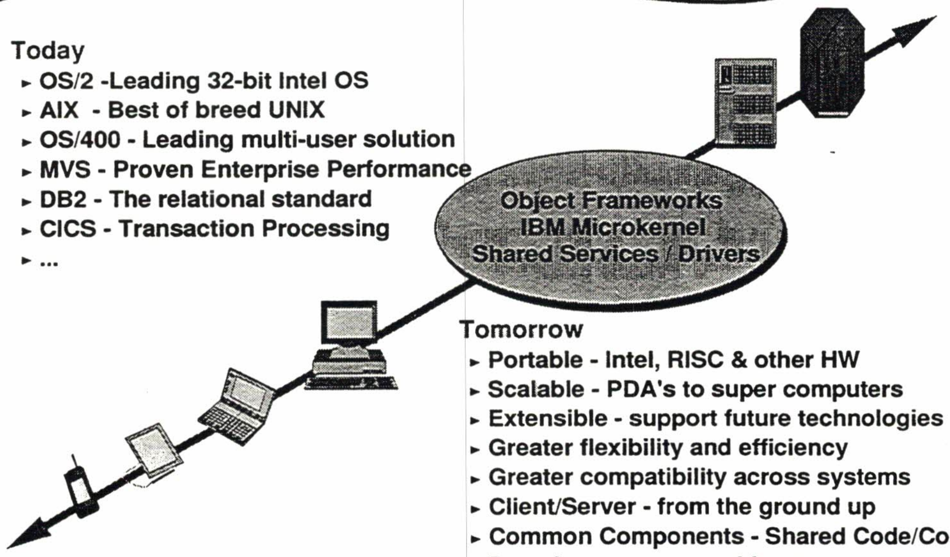
Strategy Principles:
- Scalable Solutions
- Human-Centered™ Interfaces
- Reusable Software Technologies
- Open Distributed Computing
- Investment Protection



- Embodied in a set of leading open standards and technologies
- Deployed across the entire spectrum of IBM software
- Designed to revolutionize the way software is developed and used

OS/2 WARP Portable and Scalable Software

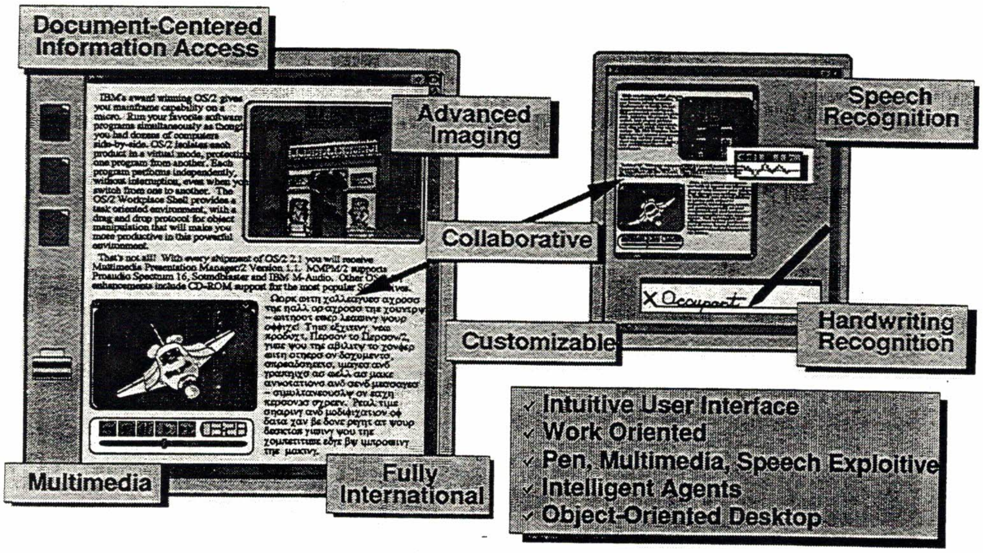
Today
- OS/2 -Leading 32-bit Intel OS
- AIX - Best of breed UNIX
- OS/400 - Leading multi-user solution
- MVS - Proven Enterprise Performance
- DB2 - The relational standard
- CICS - Transaction Processing
- ...



Object Frameworks
IBM Microkernel
Shared Services / Drivers

Tomorrow
- Portable - Intel, RISC & other HW
- Scalable - PDA's to super computers
- Extensible - support future technologies
- Greater flexibility and efficiency
- Greater compatibility across systems
- Client/Server - from the ground up
- Common Components - Shared Code/Cost
- Based on an open architecture

OS/2 WARP Human-Centered Interfaces



Document-Centered Information Access

Advanced Imaging

Collaborative

Speech Recognition

Customizable

Handwriting Recognition

Multimedia

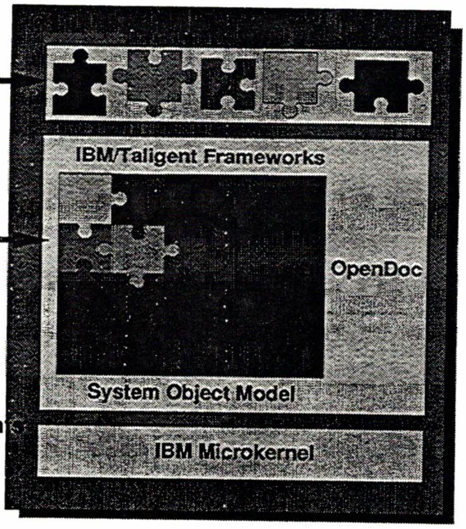
Fully International

- ✓ Intuitive User Interface
- ✓ Work Oriented
- ✓ Pen, Multimedia, Speech Exploitive
- ✓ Intelligent Agents
- ✓ Object-Oriented Desktop

OS/2 WARP Reusable Software Technologies

What you write

What you get



IBM/Taligent Frameworks

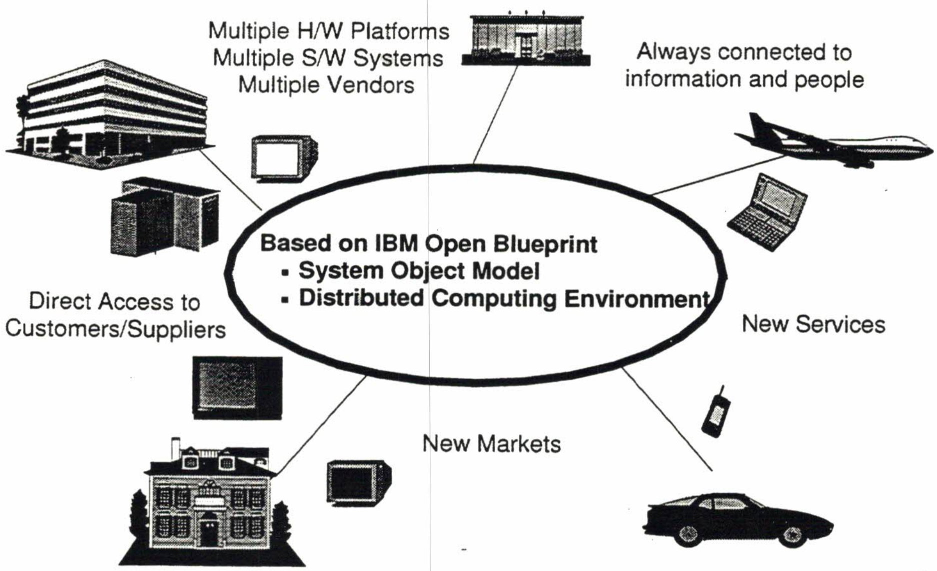
OpenDoc

System Object Model

IBM Microkernel

- Faster, more affordable development
- Extends useful life of existing systems
- Foundation for business reengineering
- Faster response to business requirements

OS/2 WARP Open Distributed Computing



Multiple H/W Platforms
Multiple S/W Systems
Multiple Vendors

Always connected to
information and people

Based on IBM Open Blueprint
- **System Object Model**
- **Distributed Computing Environment**

Direct Access to
Customers/Suppliers

New Services

New Markets

OS/2 WARP Investment Protection . . .

- ***Flexible and modular solutions***
- ***Consistency and reuse across multiple platforms***
- ***Based on open standards***
- ***Safe and reliable computing decisions***
- ***Implement at your own pace***
- ***IBM is the proven leader in legacy support***

this page intentionally left blank

MM 0301

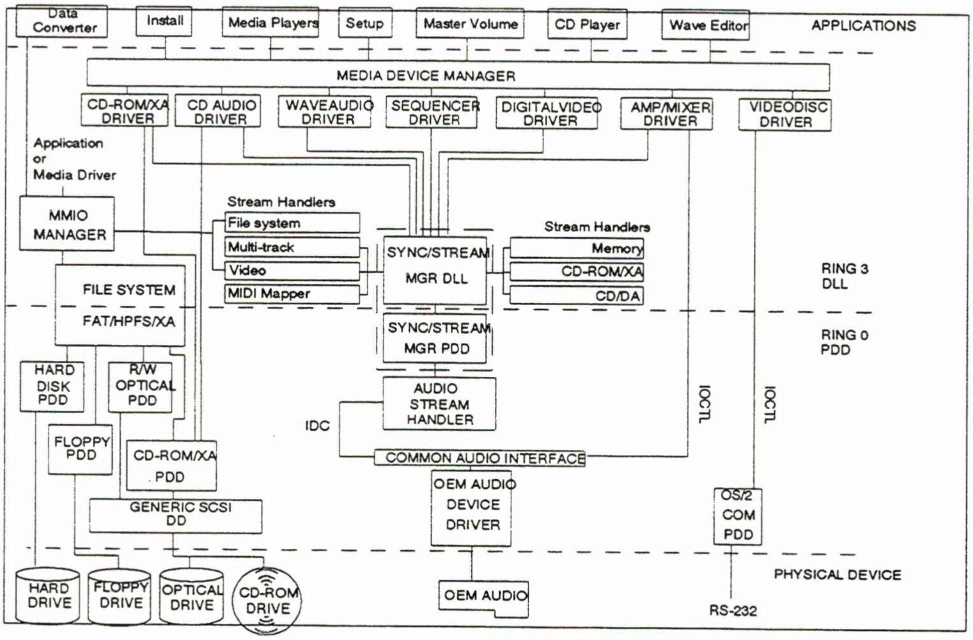
OS/2 Multimedia for Developers

Maria Ingold
IBM

Agenda

- Overview of OS/2 Multimedia architecture
- Using the Media Control Interface (MCI)
- Using Multimedia I/O (MMIO)
- Synchronization/Streaming Manager (SSM)
- OS/2 Multimedia tips and techniques
- The future of OS/2 Multimedia
- Available audio and video cards
- Useful publications
- Available forums
- Download sites
- Useful phone numbers
- Other Sessions of interest

OS/2 Multimeda Architecture


Data Converter
Install
Media Players
Setup
Master Volume
CD Player
Wave Editor
APPLICATIONS
MEDIA DEVICE MANAGER
CD-ROM/XA DRIVER
CD AUDIO DRIVER
WAVEAUDIO DRIVER
SEQUENCER DRIVER
DIGITALVIDEO DRIVER
AMP/MIXER DRIVER
VIDEODISC DRIVER
Application or Media Driver
MMIO MANAGER
Stream Handlers
File system
Multi-track
Video
MIDI Mapper
SYNC/STREAM MGR DLL
Stream Handlers
Memory
CD-ROM/XA
CD/DA
RING 3 DLL
FILE SYSTEM
FAT/HPFS/XA
SYNC/STREAM MGR PDD
RING 0 PDD
HARD DISK PDD
R/W OPTICAL PDD
AUDIO STREAM HANDLER
IOCTL
IOCTL
IDC
FLOPPY PDD
CD-ROM/XA PDD
COMMON AUDIO INTERFACE
OEM AUDIO DEVICE DRIVER
GENERIC SCSI DD
OS/2 COM PDD
PHYSICAL DEVICE
HARD DRIVE
FLOPPY DRIVE
OPTICAL DRIVE
CD-ROM DRIVE
OEM AUDIO
RS-232

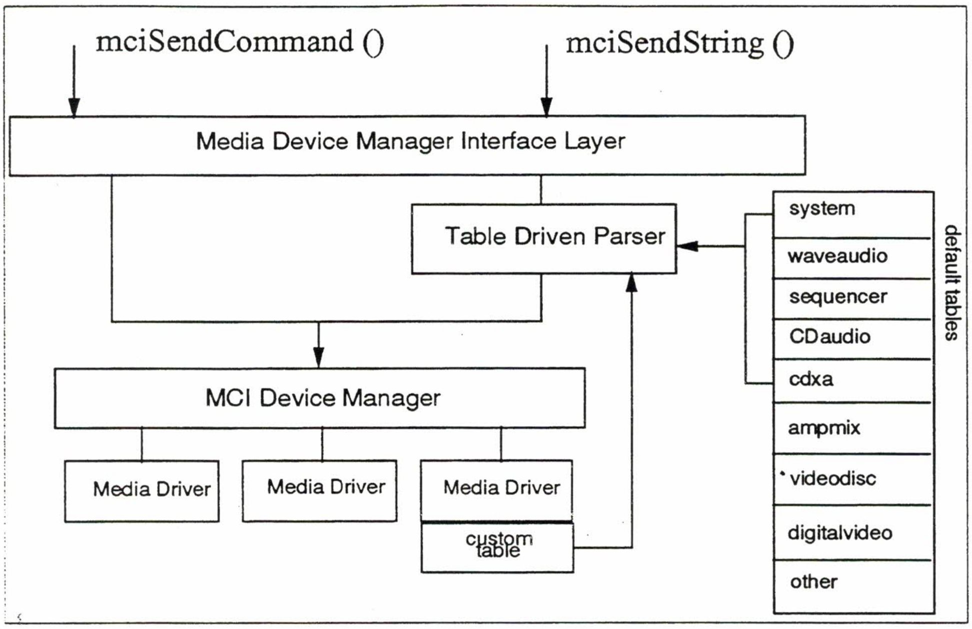
Media Control Interface

- Device independent
  - Provides apps with a logical view of MM devices in systen
- String and Procedural Interface
- Multimedia Device Resource Management
- Multimedia Device Grouping
- Timing events

MCI Command APIs

- mciSendString
  - English string interface
- mciSendCommand
  - Procedural interface
    - Message
    - Flags
    - Data structure

MCI Commands


mciSendCommand ()

mciSendString ()

Media Device Manager Interface Layer

Table Driven Parser

MCI Device Manager

Media Driver

Media Driver

Media Driver

custom table

system

waveaudio

sequencer

CDaudio

cdxa

ampmix

videodisc

digitalvideo

other

default tables

mciSendString

```
ULONG mciSendString( PSZ       pszCommand,
                     PSZ       pszReturnString,
                     USHORT usReturnLength,
                     HWND     hwndCallback,
                     UShORT usUserParm)
```

- **pszCommand:** <command> <device name> <arguments>
- **pszReturnString:** Application supplied buffer for return dat
- **usReturnLength:** Length of pszReturnStringBuffer
- **hwndCallback:** PM window handle - asynchronous notifica
- **usUserParm:** Application defined returned on notification

String Command Example

Play a Digital Audio File

```
open waveaudio alias w shareable wait
load w sound.wav wait
set w time format ms wait
setpositionadvise w on every 500 wait
play w notify
seek w to 0 wait
play w notify
close w wait
```

mciSendString - CD Example

```
#define INCL_MCIOS2
#include <os2me.h>
USHORT  usCDDeviceID;
HWND     hwndWindow;
USHORT  usUserParm;
CHAR     szReturnString[128];
CHAR     szCommand[128];
ULONG    rc;
strcpy((PSZ)szCommand, "play cd1 from 0 to 10000 notify");
usUserParm++;
rc= mciSendString( (PSZ)szCommand,
                   NNULL,
                   0L,
                   hwndWindow,
                   usUserParm);
```

mciCommandString

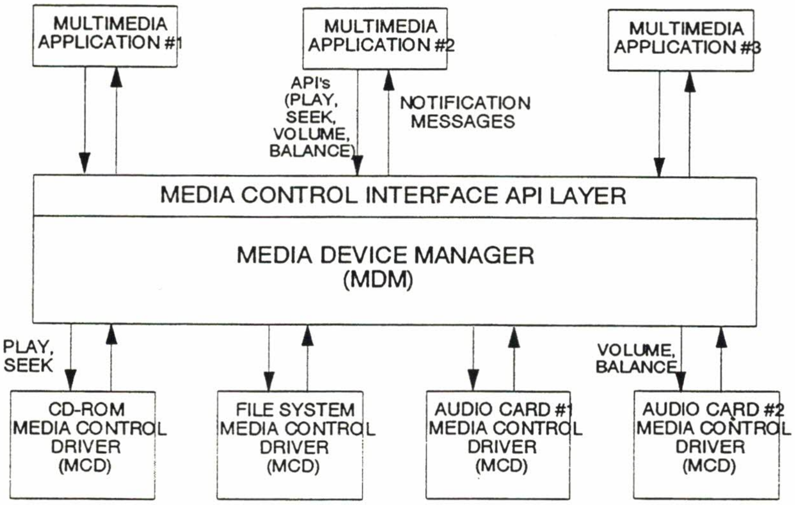
```
ULONG mciSendCommand( USHORT  usDeviceID,
                      USHORT  usMessage,
                      ULONG   ulParam1,
                      PVOID   pParam2,
                      USHORT  usUserParm)
```

- **usDeviceID**: Device ID number returned on MCI_OPEN
- **usMessage:** Command or message to execute
- **ulParam1**: Flag variable associated with usMessage
- **pParam2:** Pointer to structure associated with usMessage
- **usUserParm:** Application defined return on MM_MCINOTI

mciSendCommand - CD Example

```
#define INCL_MCIOS2
#include <os2me.h>
USHORT              usUserParm;
USHORT              usCDDeviceID;
HWND                hwndWindow;
MCI_PLAY_PARMS PlayParms;
ULONG               rc;
PlayParms.hwndCallback = hwndWindow;
PlayParms.ulFrom          = 0L;
PlayParms.ulTo            = 10000L;
UserParm++;
rc= mciSendCommand(usCDDeviceID,
                    MCI_PLAY,
                    MCI_NOTIFY | MCI_FROM | MCI_PLA
                    (PVOID)&PlayParms,
                    usUserParm);
```

MCI Command Structure



MULTIMEDIA APPLICATION #1

MULTIMEDIA APPLICATION #2

MULTIMEDIA APPLICATION #3

API's (PLAY, SEEK, VOLUME, BALANCE)

NOTIFICATION MESSAGES

MEDIA CONTROL INTERFACE API LAYER

MEDIA DEVICE MANAGER (MDM)

PLAY, SEEK

VOLUME, BALANCE

CD-ROM MEDIA CONTROL DRIVER (MCD)

FILE SYSTEM MEDIA CONTROL DRIVER (MCD)

AUDIO CARD #1 MEDIA CONTROL DRIVER (MCD)

AUDIO CARD #2 MEDIA CONTROL DRIVER (MCD)

High Level MCI APIs

- mciPlayFile
  - Simplified method to play multimedia data file
  - Synchronous or asynchronous mode
  - Provide display window handle for video playback
  - Default audio or video device used for playback

High Level MCI APIs

- mciPlayResource
  - Plays a program resource
  - Name of data file to play is retrieved from program resourc
  - Data file opened using MMIO
  - Resource types
    - RT_WAVE: Digital Audio
    - RT_AVI: Digital Video
    - RT_RMID: MIDI
    - RT_RIFF: RIFF

High Level MCI APIs

- mciRecordAudioFile
  - Simplified method to record an audio file
  - Can record MMIO compound audio file element
  - Only records in 11kHz, mono, PCM

REXX Programming Interfaces

- REXX APIs
  - mciRxSendString
    - Access to mciSendString API
  - mciRxGetErrorString
    - Access to mciGetErrorString API
  - mciRxGetDeviceID
    - Access to mciGetDeviceID
  - mciRxInit
  - mciRxExit
- Limitations
  - Cannot receive PM messages, so no MMPM notifications
  - Should adhere to following rules
    - Issue **wait** keyword
    - Open device **shareable** and acquire **exclusive instanc**
    - If need PM session use PMREXX

REXX Programming Example - I

```
rc= RXFUNCADD( 'mciRxInit', 'MCIAPI', 'mciRxINIT')
rc = mciRxInit()
MCICommand = 'open bell.wav alias wavealias wait shareabl
rc= mciRxSendString(MCICommand, 'ReturnString', '0', '0')
if rc <> 0 then
  do Errorrc=rc
      rc = mciRxGetErrorString(Errorrc, 'ErrorString')
  end
else
  do
  MCICommand = 'acquire wavealias exclusive instance wait
  rc = mciRxSendString(MCICommand, 'ReturnString', '0', '0'
  if rc <> 0 then
    do Errorrc = rc
    rc = mciRxGetErrorString( Errorrc, 'ErrorString')
    end
```

REXX Programming Example - II

```
else
  do MCICommand = 'play wavealias wait'
  rc = mciRxSendString( MCICommand, 'ReturnString', '0', '0
  if rc <> 0 then
     do Errorrc = rc
     rc = mciRxGetErrorString( Errorrc, 'ErrorString')
     end
  else
     do MCICommand = 'close wavealias wait'
     rc = mciRxSendString(MCICommand, 'ReturnString', '0',
     end
   end
rc = mciRxExit()
```

Device Resource Management

- Attempts to fully utilze resources of the multimedia devices
- Device instances can be one of three sharing states
  - Fully shareable - least restrictive
    - **shareable** flag
    - Make device instance inactive for other to become activ
    - Other device executes **open** or **acquire**
  - Exclusive instance
    - **exclusive instance** flag
    - Device instance cannot be made inactive (at this time)
    - For example during a **record** or **set**
  - Exclusive - most restrictive
    - **exclusive** flag
    - Entire media device is held exclusively
    - Default state on **open** command

Passing the Device

```
case MM_MCIPASSDEVICE:
   usDeviceID = (USHORT) SHORT1FROMMP( mp1);
   if (SHORT1FROMMP(mp2) == MCI_GAINING_USE)
      /* GAINING USE */
   else
      /* LOSING USE */
break;
```

- MM_MCIPASSDEVICE message
  - MP1
    - usDeviceID (USHORT): Media device instance ID numb
    - usReserved (USHORT): Unused
  - MP2
    - usEvent (USHORT): Media device instance active/inacti
    - usReserved (USHORT): Unused

Device Grouping

- Multiple device instances logically grouped together
  - Control more than one device at a time
- **piecemeal** (default)
  - State of individual device instances are independent
- **nopiecemeal**
  - All device instances in group in same device state

Device Group Commands

- **acquire**
- **release**
- **play**
- **record**
- **seek**
- **close**
- **stop**
- **pause**
- **resume**

Device Group Example

```
open cdaudio alias cd1 shareable wait
open music.wav alias myaudio shareable wait
group mygroup make nopiecemeal (cd1 myaudio) wait
acquire mygroup exclusive instance wait
```

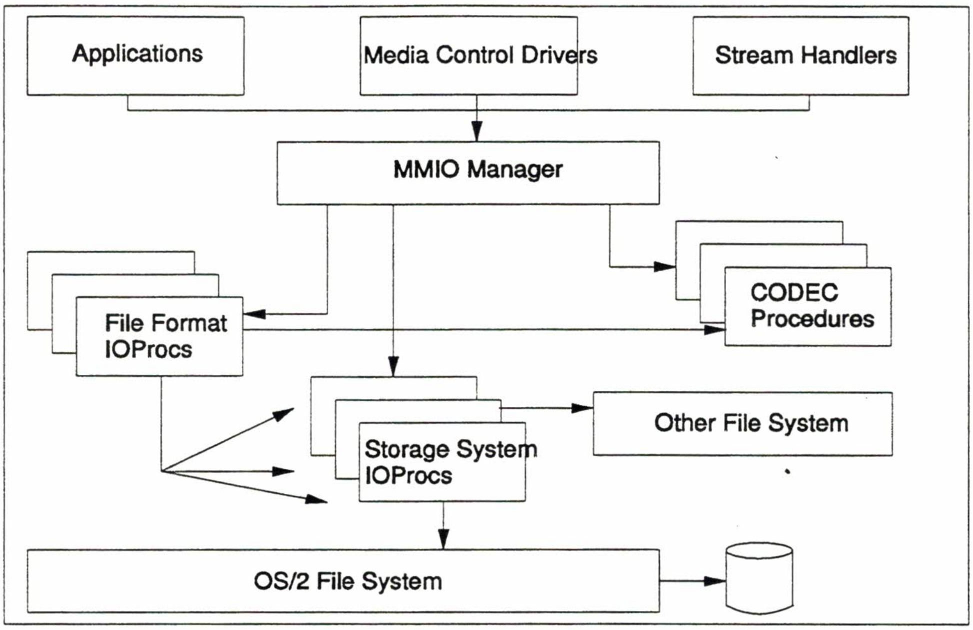
Timing Events

- **setpositionadvise**
  - Continuous timing events
  - Based on current playback or recording time
  - Update internal counters or external user interface
  - One periodic notification message
    - **MM_MCIPOSITIONCHANGE**
- **setcuepoint**
  - One-time notification
  - Based on current playback or recording time
  - Used to perform an action at a specific time
  - Returns message to application at a set time
    - **MM_MCICUEPOINT**

Multimedia I/O (MMIO)

- Extendable with I/O Procs
- Data independece
  - Storage systems
  - File formats
  - CODEC management
- Data identification
- Data conversion
- Internal RIFF support

MMIO Architecture



Applications

Media Control Drivers

Stream Handlers

MMIO Manager

File Format IOProcs

CODEC Procedures

Storage System IOProcs

Other File System

OS/2 File System

mmioOpen and mmioClose APIs

```
HMMIO mmioOpen(PSZ pszFileName,
               PMMIOINFO pmmioinfo,
               ULONG ulOpenFlags)
```

- **pszFilename:** File or element in bundle file to open
- **pmmioinfo:** Structure containing extra parameters
- **ulOpenFlags:** Flags that define how the file is used for ope

```
USHORT mmioClose(HMMIO hmmio,
          USHORT usFlags)
```

- **hmmio:** Open file handle returned by mmioOpen
- **usFlags:** Nothing or MMIO-FHOPEN - FOURCC_DOS files

MMIO Open Flags

- **MMIO_READ**
- **MMIO_WRITE**
- **MMIO_READWRITE**
- **MMIO_BUFFSHARED**
- **MMIO_VERTBAR**
- **MMIO_EXCLUSIVE**
- **MMIO_DENYWRITE**
- **MMIO_DENYREAD**
- **MMIO_DENYNONE**
- **MMIO_CREATE**
- **MMIO_DELETE**
- **MMIO_ALLOCBUF**
- **MMIO_APPEND**
- **MMIO_NOIDENTIFY**

mmioOpen Example

```
#define         DATA_SIZE 1024
MMIOINFO        mmioinfo;
HMMIO           hmmioMem;
HAB             habClipboard;

mmioinfo.fccIOProc        = mmioFourcc( 'W', 'A', 'V', 'E' );
mmioinfo.fccChildIOProc = FOURCC_MEM;
mmioinfo.cchBuffer        = DATA_SIZE;
mmioinfo.pchBuffer        = (PCHAR) WinQueryClipbrdData( habClipboard,
                                                         CF_WAVE );

hmmioMem                  = mmioOpen( NULL,
                                      &mmioinfo,
                                      MMIO_READ );
...

mmioClose( hmmioMem );
```

mmioSendMessage API

```
LONG mmioSendMessage(HMMIO hmmio,
                     USHORT usMsg,
                     LONG Param1,
                     LONG Param2)
```

- **hmmio:** Open file handle returned by mmioOpen
- **usMsg:** A MMIO_M... predefined or user-defined messag
- **Param1:** Additional message information
- **Param2:** Additional message information

Applications use mmioSendMessage to communicate directly to an I/O Procedure.

I/O Procedures

- Two types
  - File format
    - **MMIO_IOPROC_FILEFORMAT**
    - Allows identification and manipulation of file format data
    - Digital audio, bitmap image, digital video, MIDI,text,...
    - MMIO hides details of each file format
    - Allows comman access across each format type
    - Includes support for RIFF
  - Storage system
    - **MMIO_IOPROC_STORAGESYSTEM**
    - Unwraps data object for use by file format I/O proc
- Identified by a FOURCC

Using I/O Procedures

- Basic APIs
  - mmioOpen
  - mmioClose
  - mmioIdentifyFile
  - mmioGetFormatName
  - mmioGetFormats
  - mmioGetHeader
  - mmioQueryHeaderLength
- Equivalent basic messages
  - MMIOM_OPEN
  - MMIOM_CLOSE
  - MMIOM_IDENTIFYFILE
  - MMIOM_GETFORMATNAME
  - MMIOM_GETFORMATINFO
  - MMIOM_GETHEADER
  - MMIOM_QUERYHEADERLENGTH

Using I/O Procedures - Example

```
RC                    rc;
HMMIO                 hmmioMem;
LONG                  lBytesRead;
MMAUDIOHEADER         mmaudioheader;
UCHAR                 *pBuffer;
ULONG                 ulBufLen ;
rc = mmioGetHeader( hmmioMem,
                      (PVOID) &mmaudioheader,
                      sizeof( mmaudioheader ),
                      &lBytesRead,
                      (ULONG) NULL,
                      (ULONG) NULL );
ulBufLen = mmaudioheader.mmXWAVHeader.
                        XWAVHeaderInfo.ulAudioLengthInBytes;
pBuffer = malloc( ulBufLen );
rc = mmioRead( hmmioMem,
                 (PSZ) *pBuffer,
                 ulBufLen );
```

Audio I/O Procedures & FOURCC

Apple IFF Wave Digital Audio	AIFF
Amiga IFF Format	IFF
Creative Labs Inc. Voice Audio	VOC
IBM AVC ADPCM Digital Audio	AVCA
RIFF Waveform Digital Audio	WAVE
Unix (NeXT/SUN) SND Format	SND
Musical Instrument Digital Interface (MIDI)	MIDI, RMID

Image I/O Procedures & FOURCC

GIF Image Compressed	GIFC
IBM AVC Still Video Image	AVCI
IBM MMotion Still Video Image	MMOT
IBM OS/2 1.3 Bitmap	OS13
IBM OS/2 2.0 Bitmap	OS20
Kodak PhotoCD	PCD
MS Windows Device-Independent Bitmap (DIB)	W130
PCX (PC Paintbrush) Image Compressed	PCXC
RIFF Device-Independent Bitmap (RDIB)	RDIB
Targa Image Compressed	TGAC
Targa Image Uncompressed	TGAU
TIFF FAX Compressed	TFFC
TIFF Intel Compressed	TFIC
TIFF Intel Uncompressed	TFIU
TIFF Motorola Compressed	TFMC
TIFF Motorola Uncompressed	TFMU

Installing an I/O Procedure

- Three methods
  - Temporary install
    - mmioOpen
  - Semi-permanent install into MMIO manager table
    - mmioInstallIOProc
  - Permanently install into the mMPMMMIO.INI file
    - mmioIniFileHandler

CODEC Procedures

- Two types of CODECs - Compression/Decompression
  - Compressor CODECs
  - Decompressor CODECs
- CODEC can be called directly by application
- CODEC can be called indirectly through file format I/P Pro
  - MMIOM_COMPRESS
  - MMIOM_DECOMPRESS
- Can be a hardware CODEC or a software CODEC

Digital Video CODECs & FOURCC

Ultimotion	ULTI
Indeo 2.1	RT21 rt21
Indeo 3.1	IV31 iv31
Indeo 3.2	IV32 iv32
FLC/FLI	flic
MPEG-1	MPEG
RAW	DIB

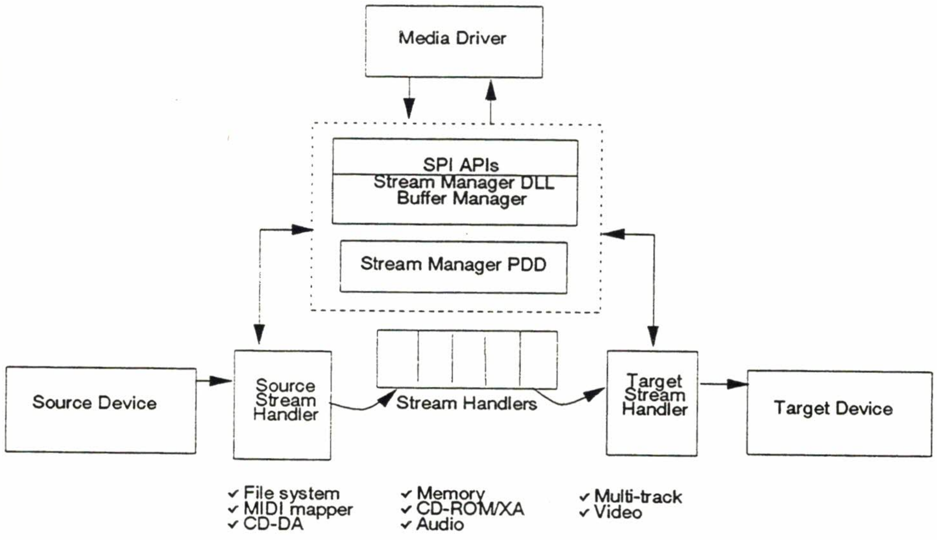
CODEC Messages

- MMIOM_CODEC_OPEN
- MMIOM_CODEC_CLOSE
- MMIOM_CODEC_QUERYNAME
- MMIOM_CODEC_QUERYNAMELENGTH
- MMIOM_CODEC_COMPRESS
- MMIOM_DECOMPRESS

Synch/Streaming Manager

- Centralized management of resources
- Centralized distribution of timing information
- Extendable architecture with stream handlers
- Synchronization
- Efficient data movement
- Stream time tolerance checking

Synch/Stream Architecture



Media Driver

SPI APIs
Stream Manager DLL
Buffer Manager

Stream Manager PDD

Source Device

Source Stream Handler

Stream Handlers

Target Stream Handler

Target Device

✓ File system
✓ MIDI mapper
✓ CD-DA

✓ Memory
✓ CD-ROM/XA
✓ Audio

✓ Multi-track
✓ Video

Tips and Techniques

- Media Control Interface
  - To improve mciStringCommandParsing
    - issue string commands in lower case
  - Use a synchronous **wait** flag when
    - Commands are processed quickly
  - Use an asynchronous **notify** flag when
    - Commands take a long or indefinite period of time
  - Setpositionadvise message
    - The more frequent the message, the bigger the overhe
- General
  - Minimum 16k stack size. Preferred is 32 to 64k.
  - Spawn a separate thread to process window messages
  - For multiple files, **open** the device once and **load** other fi
  - Remember that maximum number of file handles is 100

Audio Cards

- Ultrasound Ace by Advanced Gravis
- Ultrasound by Advanced Gravis
- Ultrasound Max by Advanced Gravis
- Sound Galaxy Nova 16 Extra by Aztech Systems Ltd.
- Sound Blaster (16, Pro, AWE-32) by Creative Labs
- Portable Sound Plus by DSP Solutions
- AudioDrive by ESS Technology Inc.
- Audiovation by IBM
- M-Audio Capture and Playback Adaptor by IBM
- Business Audio (AD1848 Chip) by IBM
- Sound Man 16 by Logitech
- Pro Audio Spectrum 16 by Media Vision
- Pro Audio Studio by Media Vision
- Jazz 16 by Media Vision
- WinSound by Sigma Designs

Audio Cards - Continued

- AD 1848 (Business Audio) by Analog Devices
- Plug-N-Play by Arkay Technologies
- CS4321/CS4248 by Crystal Semiconductor
- Audio Drive 488 by ESS Technologiy
- Audio Drive 688 by ESS Technology
- ISP 16 by Media Magic
- SC500 by Reveal
- Sound Expression 14.4 VSp by Boca Research
- Mozart 16 by Oak

Video Cards - Capture

- Video Capture Adapter by IBM
- SuperVIA by Jovian Logic
- QuickVIA by Jovian Logic
- M&M Basic by Omnicorp Graphics Corporation
- Super Video Windows by New Media Graphics
- Video Blaster by Creative Labs
- Video Blaster FS200 by Creative Labs
- Video Clipper by CEI
- Video Magic by Samsung
- WaveWatcher by AITech
- WinMovie by Sigma Design
- Illuminator-16 by Matrox
- Marvel II by Matrox
- MediaBurst Movie by IBM
- V7320 Video Controller by ASCII Corporation

Video Cards - Playback

- Video
  - V7320 Video Controller by ASCII Corporation
  - ReelMagic by Sigma Designs
  - ReelMagic Lite by Sigma Designs
- Television
  - Win/TV by Hauppauge
  - Win/TV Celebrity by Hauppage
  - PC/TV by Tecmar

OS/2 Multimedia Publications

- Articles
  - Digital video architecture for OS/2 2.1
    - *High -Speed Networking and Multimedia Computing*
    - ISBN: 0-8194-1483-2
  - Inside OS/2 Software Motion Video
    - Dr. Dobb's Multimedia Sourcebook, #209
- Chapters
  - Ch 15, Multimedia and OS/2
    - *Dvorak's Guide to OS/2*
  - Ch 15, Multimedia
    - *OS/2 2.11 Unleashed*

OS/2 Multimedia Publications

- Books
  - Multimedia Application Programming Guide
    - G25H-7102
  - Multimedia Programming Reference
    - G25H-7114
  - Multimedia Subsystem Programming Guide
    - G25H-7113
    - Ultimedia Video IN Programming Reference
    - S82G-4225
  - Can order the above books in US: (800) 342-6672
    - In Canada: (800) 465-4234
  - Developing OS/2 Multimedia Applications
    - ISBN: 0-442-01929-7

OS/2 Multimedia Publications

- Red Books (with sample diskettes)
  - MMPM/2 Advanced Programming Techniques
    - GG24-4042
  - MMPM/2 Programming Using Digital Audio Examples
    - GG24-3963
- CD ROMs
  - Developer Connection
    - wwdap@vnet.ibm.com
    - (407) 982-6408
  - Ultimotion Development Kit

OS/2 Multimedia Forums

- Internet
  - comp.os.os2.multimedia
  - comp.multimedia
- America Online
  - keyword OS2
- Compuserve
  - OS2SUPPORT (Various sections)
  - OS2USER (Various sections)
  - OS2DF1 FORUM, Section 12
  - GO IBM
  - FIND OS2
- Prodigy
  - jump OS/2 CLUB
- World Wide Web
  - http://www.austin.ibm.com/pspinfo/os2.html

Download Sites

- America Online
  - keyword OS2
- Compuserve
  - OS2DF1, Section 12
- Prodigy
  - jump OS/2 CLUB
- FTP
  - hobbes.nmsu.edu
  - ftp-os2.cdrom.com
    - /pub/os2/32bit/multimedia

BBSs - USA

Fernwood	(203) 483-0348
The Bin BBS	(206) 451-1905
OS/2 Source BBS	(303) 744-0373
Denver OS/2 BBS	(303) 755-6859
Inside Technologies BBS	(313) 283-1511
OS/2 Woodmeister	(314) 446-0016
Gateway/2	(314) 554-9313
Pyramid/2	(415) 494-7497
Bay Area OS/2	(510) 657-7948
OS/2 San Diego	(619) 558-9475
OS/2 Las Vegas	(702) 433-5535
OS/2 Shareware	(703) 385-4325
Greater Chicago Online	(708) 895-4042
OS/2 Exchange BBS	(904) 739-2445
Information about US OS/2 BBSs:	(800) 547-1283

BBSs - Europe, Australia, Canada

Abaforum (Barcelona)	34-3-589.38.88
IBM UK	44-256-336655
OS/2 UK	44-454-633197
IBM Denmark	45-42-88-72-22
Copenhagen UG BBS	45-48-24-45-80
OS/2 Norway	47-66-99-94-50
IBM Norway	47-66-99-94-50
IBM Germany	49-7034-15-2160
OS/2 Australia	61-2-241-2466
Soft/2 Shareware (Adelaide)	61-8-370-7339
IBM Canada	(416) 946-4244
IBM Canada	(416) 946-4255
IBM Canada	(514) 938-3022
IBM Canada	(604) 380-5411
IBM Canada	(604) 664-6464

Useful Phone Numbers

■ Audio Card Manufacturers	
• Advanced Gravis	BBS (604) 431-5927
• Arkay Technologies	Tel (603) 425-2149
• Aztech Systems Ltd.	BBS (501) 623-8933
• DSP Solutions	BBS (415) 494-1621
• Media Vision	BBS (510) 770-0527
■ Video Cards	
• Matrox	Tel (514) 685-2630
• Omnicomp Graphics	Tel (514) 685-2630
• Tecmar	Tel (800) 422-2587
■ IBM	
• (800) 3 IBM OS2 (3 426 672)	

Other Sessions of Interest

- For Users
  - OPPS11 Harnessing the Power of the Warp BonusPak
  - DDOS28 OS/2 Multimedia User Tips and Techniques
- For Developers
  - MMOS02 OS/2 Multimedia for Developers
  - DDOS18 Video Playback and Capture Under OS/2
  - DDOS24 Multimedia DD Overview and Future Directions
  - DDOS25 Multimedia Audio Device Drivers
  - DDOS26 Multimedia Video Capture Device Drivers
  - DDOS27 MPEG Playback Device Drivers
  - DDOS29 Multimedia Device Driver Installation and Test
  - PPOS05 Natural Computing on OS/2 for the PowerPC

MNC 0302

Installing the Netware Client for OS/2

Bob Panza

The Systems Expert, Inc.
Woodbury, Conn.

SYNOPSIS

A "live" installation, configuration, and general review of the NetWare Client for OS/2 will focus on the latest version (NetWare Client, v2.1 and OS/2, v2.11 at the time of this writing). The session is part of a series on the NetWare Client for OS/2 which includes #086, #093, and #101.

AGENDA

1. Examination of source floppies
2. Installation process flowchart review
3. Deciding on NetWare session support options for OS/2, Windows, and DOS
4. Installation process walkthrough
5. Choosing LAN drivers
6. Configuring
7. Completing installation
8. CONFIG.SYS entries
9. NET.CFG option overview
10. ODINSUP
11. OS/2 utilities
12. Online documentation
13. Other NetWare client folder options
14. Multiprotocol/interoperability introduction

INSTALLING THE NETWARE CLIENT FOR OS/2

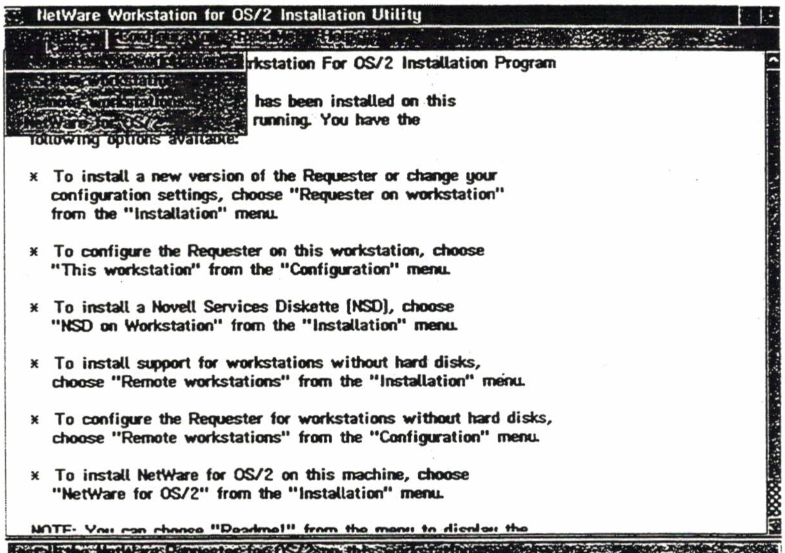
Since the initial release of OS/2 (version 1.0), the operating system transformed many times. Versions 1.1, 1.2, 1.3, and 2.0 led to the latest and greatest - version 2.1. Like evolving butterflies emerging from their cocoons over and over again, the new OS/2 versions got better with every transformation. OS/2, version 2.1 is particularly a milestone of stability and functionality.

Throughout the course of OS/2 evolution, which began in 1987, many sets of CSDs (Corrective Service Diskettes) were applied to remedy bugs and design related problems. The quantity of cumulative CSD floppies must have numbered several hundred over time. Those CSDs brought additional updates, further altering the software between released versions. Creation of the OS/2 operating system obviously turned out to be a bigger and more complex task than the original planners imagined. The miscalculation is understandable. OS/2 was a new kind of operating system, characterized by greater capability and complexity than DOS but more user friendly and less complicated in comparison to Unix.

Novell's NetWare Requester for OS/2 seemed to be forever chasing the transformations of OS/2. Connectivity with the Novell world is especially important for broad based appeal to the vast community of NetWare end users. Like OS/2, the NetWare Requester has travelled a long road through many releases before version 2.01. The parallel holds for software fixes as well, with Novell issuing NSDs (Novell Service Diskettes) to remedy the numerous bugs which all new software is fated to have.

Metamorphoses

Even the name of version 2.01 changed from "NetWare Requester for OS/2" to "NetWare Workstation for OS/2". However, the term has not yet been altered in the software and documentation details so there are many remaining references to "requester". "NetWare Requester" rather than "NetWare Workstation" is still an option in the NetWare Open Data-Link Interface (ODI) configuration file, NET.CFG, as we shall see later on. Version



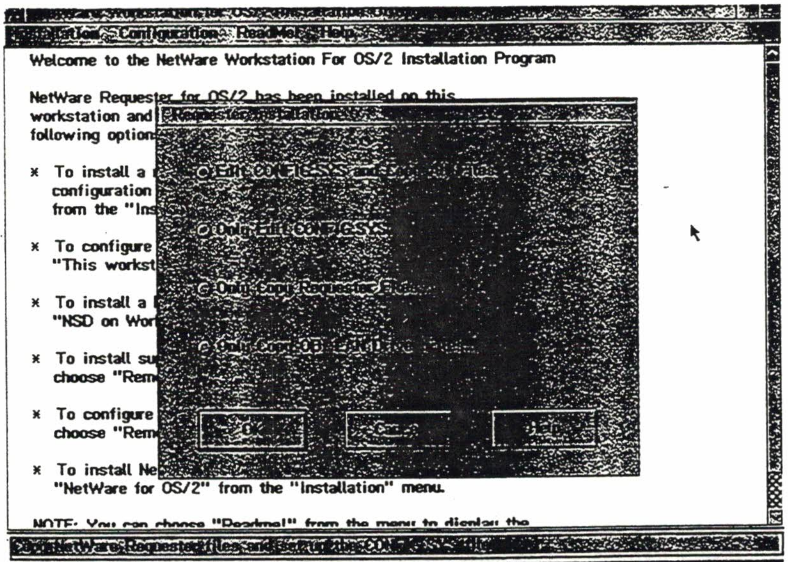
rkstation For OS/2 Installation Program

has been installed on this
running. You have the
following options available:

- To install a new version of the Requester or change your configuration settings, choose "Requester on workstation" from the "Installation" menu.
- To configure the Requester on this workstation, choose "This workstation" from the "Configuration" menu.
- To install a Novell Services Diskette (NSD), choose "NSD on Workstation" from the "Installation" menu.
- To install support for workstations without hard disks, choose "Remote workstations" from the "Installation" menu.
- To configure the Requester for workstations without hard disks, choose "Remote workstations" from the "Configuration" menu.
- To install NetWare for OS/2 on this machine, choose "NetWare for OS/2" from the "Installation" menu.

NOTE: You can choose "Readme!" from the menu to display the

Figure 1: The NetWare Client for OS/2 Installation Utility provides plenty of information to guide the installer through software installation and configuration. Application of NSDs via the utility is a nice touch. Install also covers the NetWare for OS/2 file-server product.



Welcome to the NetWare Workstation For OS/2 Installation Program

NetWare Requester for OS/2 has been installed on this
workstation and
following option

- To install a configuration from the "Ins
- To configure "This workst
- To install a "NSD on Wor
- To install su choose "Rem
- To configure choose "Rem
- To install Ne "NetWare for OS/2" from the "Installation" menu.

NOTE: You can choose "Readme!" from the menu to display the

Figure 2: Full and partial software installation options are provided as well as *config.sys* editing. The process can be run over and over again without doing damage. Afterwards, the utility remains on hand in the Novell Icon View.

2.01 of the NetWare Workstation for OS/2 is the version we installed on OS/2, version 2.1. The combination brings a incredible variety of concurrent possibilities to the OS/2 environment.

The workstation can simultaneously run a combination of OS/2, virtual DOS, virtual Windows, and real DOS kernel sessions. It is also possible to have sessions running different versions of DOS. NetWare 3.11 DOS NETX and NetWare

4.X DOS workstation Virtual Loadable Modules (VLMs) are both supported, although there are some limitations on NetWare 4.X functionality for DOS and Windows. Global or private options may be configured across the board or session by session for seven possible session types: OS/2, private virtual DOS, global virtual DOS, private virtual Windows, global virtual Windows, private sessions booted from a real DOS kernel and running NetWare 4.0 DOS workstation VLMs, and global sessions booted from a real DOS kernel running NETX.

Compatibility covers NetWare 3.1X as well as NetWare 4.X servers, again with a few caveats for DOS and Windows. Initialized software can mix ODI with Network Driver Interface Specification (NDIS) and IBM Extended Services architectures (via ODINSUP.SYS and LANSUP.SYS drivers). As a result, all popular protocol stacks may be deployed for full interoperability across an incredible range of system platforms. Full compatibility has been achieved with IBM LAN Services right down to IBM's NetBIOS/LLC and Novell's NetBIOS/IPX side by side. Most amazing of all, the workstation may run NetWare for OS/2 as a nondedicated NetWare file server along with the NetWare Workstation for OS/2! Both file server and workstation may even share the same NIC. The configuration possibilities go on and on, seeming more prolific than the Roman gods of Ovid.

In order to get the NetWare Workstation up and running on OS/2, the software must be procured, installed, and configured. Configuration involves CONFIG.SYS and NET.CFG entries to unlock workstation hardware, OS/2 communication interfaces, communication protocols, and so forth. In addition, configuration after initial installation is key for tranforming the many potentials of the IBM-Novell OS/2 environment into actualities at the whim of the workstation user.

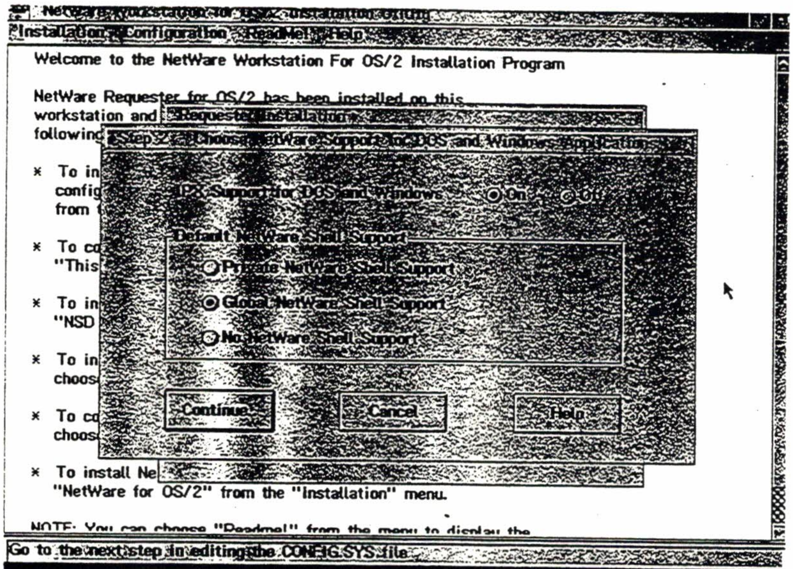
Prepping

Preparation for the OS/2 odyssey to NetWare begins with procurement of the NetWare Workstation for OS/2, version 2.01 and an examination of the software before installation. The software package and documentation may be purchased from Novell for $30 or downloaded from NetWire.

After installing IBM OS/2, v2.1 on the "guinea pig" workstation, we downloaded three self-extracting compressed files from NetWire: WSOS21.EXE, WSOS22.EXE, and WSOS2D.EXE. Each file explodes for creation of separate floppies which we respectively labelled WSOS2_1, WSOS2-2, and WSDRV-1. (It is best to explode the files on a hard disk and then XCOPY them to create the WSOS2 diskettes.) Most of the exploded files reside in subdirectories with many OS/2 files stored in packed format.

What does each floppy contain? WSOS2-1 has an installation program, INSTALL.EXE, and an unpack program, NWUNPACK.EXE, in the root directory. \LOGIN, \NLS, \WINDOWS, \DOS.NP, and \SQL directories each hold their respective files. \OS2 and \PSFONTS are also included as empty directories (without any files), most likely to build a set directory structure on the workstation. A few utility files (CX.EXE, LOGIN.EXE, MAP.EXE, and NLIST.EXE) reside packed in \LOGIN. In addition, \LOGIN holds various unicode files for language support of NetWare utilities. The unicode files are identified by "UNI" as part of their file names. National Language Support (English) and OS/2 NetWare Tools are stored below \NLS.

The Named Pipes Extender, DOSNP.EXE, which enables DOS workstations to communicate via Named Pipes with prod-


NetWare Workstation for OS/2 Installation Utility
Installation Configuration ReadMe! Help
Requester Installation
Continue
Cancel
Help

Welcome to the NetWare Workstation For OS/2 Installation Program

NetWare Requester for OS/2 has been installed on this workstation and ... following ...

* To in... config... from ...

* To co... "This ...

* To in... "NSD ...

* To in... choos...

* To co... choos...

* To install Ne... "NetWare for OS/2" from the "Installation" menu.

NOTE: You can choose "Readme!" from the menu to display the

Step 2 - Choose NetWare Support for DOS and Windows Applications

IPX Support for DOS and Windows ○ On ○ Off

Default NetWare Shell Support

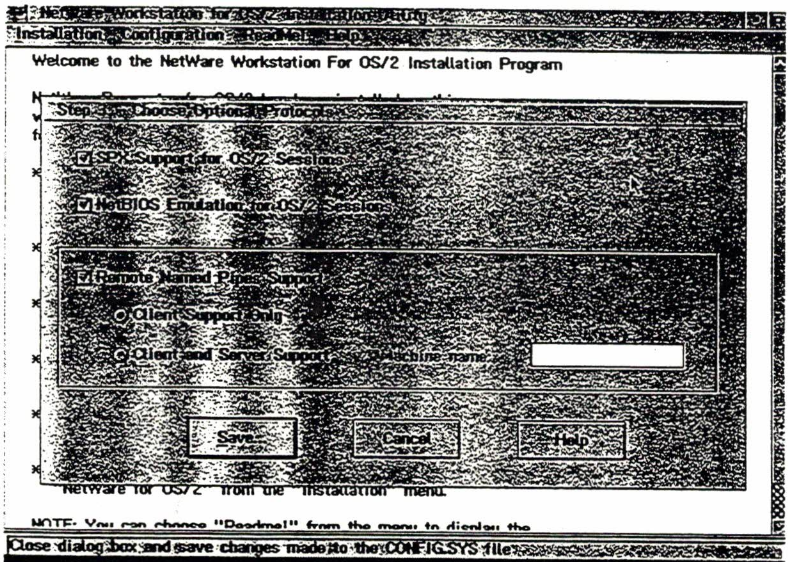
○ Private NetWare Shell Support

● Global NetWare Shell Support

○ No NetWare Shell Support

Go to the next step in editing the CONFIG.SYS file

Figure 3: IPX support for DOS and Windows brings a "better Windows than Windows" to the network. Global NetWare shell support is the best way to start out. Even with the global default in place, private sessions can be set up individually.


NetWare Workstation for OS/2 Installation Utility
Installation Configuration ReadMe! Help
Save
Cancel
Help

Welcome to the NetWare Workstation For OS/2 Installation Program

NetWare for OS/2" from the "Installation" menu.

NOTE: You can choose "Readme!" from the menu to display the

Step 3 - Choose Optional Protocols

☑ SPX Support for OS/2 Sessions

☑ NetBIOS Emulation for OS/2 Sessions

☑ Remote Named Pipes Support

○ Client Support Only

○ Client and Server Support Machine name

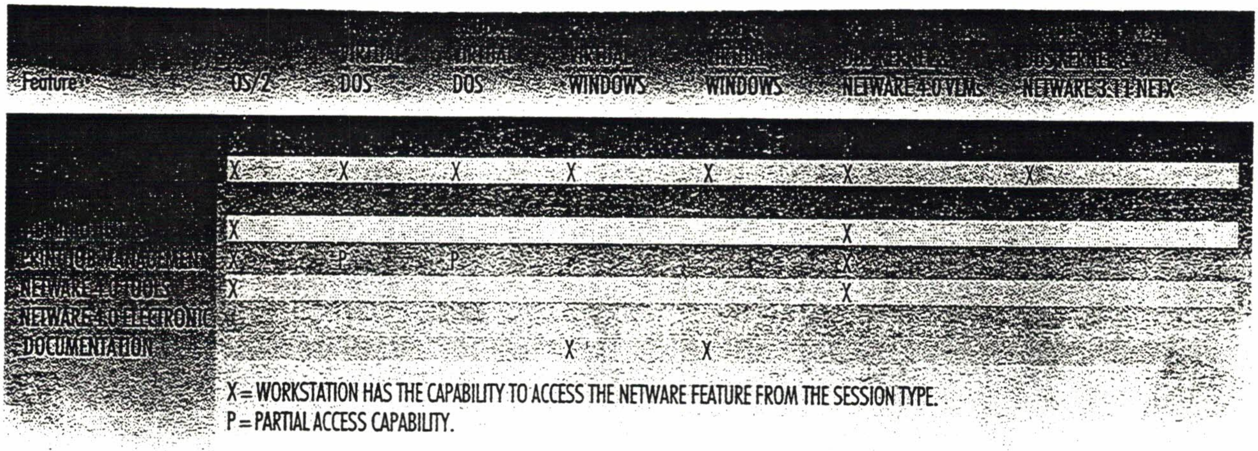
Close dialog box and save changes made to the CONFIG.SYS file

Figure 4: Optional protocols reflect the Novell tradition of SPX/IPX, NetBIOS emulation/IPX, and Named Pipes/SPX. Named Pipes client and server support and machine name creates the Named Pipes server daemon platform for a Named Pipes client/server backend. Install will promptly update the *config.sys* file based on the selections.

ucts such as OS/2-based Microsoft SQL Server, resides in \DOS.NP. NETAPI.DLL and NETOEM.DLL, for SQL Server clients running Microsoft Windows, are found in the \SQL subdirectory. The tradition continues whereby those key DOS and Windows workstation files are bundled with NetWare Workstation for OS/2. They must be manually copied from WSOS2-1 onto the DOS or Windows workstations which require them. The Named Pipes NETAPI.DLL in \SQL should not be confused with the OS/2 NETAPI.DLL in \REQUESTR or the WIN-OS/2 NETAPI.DLL in \WINDOWS on WSOS2_1. Each NETAPI serves a different purpose.

WSOS2_2 stores a README.1ST file and a \REQUESTR subdirectory containing the OS/2 workstation files, such as LSL.SYS, IPX.SYS, ODINSUP.SYS, NWREQ.SYS, VSHELL.SYS, etc. The readme file provides valuable configuration related information for the installer. In \OS2, floppy WSDRV_1 holds the LAN drivers for accessing different network interface cards (NICs) or higher level industry standard interfaces provided by software such as IBM Extended Services.

Before running the installation program, we reviewed the readme file with an ASCII file editor. This is an important practice when working with NetWare software. Last minute information vital to making the software work properly is sometimes found in the documentation files rather than the manual. In advance of installation and configuration, we learned a few important things that could impact our configuration related decisions. We read that remote workstations are not yet supported; the Remote Initial Program Load files were promised for an unspecified later date. The new Ethernet default for IPX under ODI drivers is the 802.2 frame type. This is consistent with NetWare 4.X, and with new



Feature	OS/2	VIRTUAL DOS	VIRTUAL DOS	VIRTUAL WINDOWS	VIRTUAL WINDOWS	DOS KERNEL NETWARE 4.0 VLMS	DOS KERNEL NETWARE 3.11 NETX
[illegible]	X	X	X	X	X	X	X
[illegible]	X					X	
PRINT JOB MANAGEMENT	X	P	P			X	
NETWARE 4.0 TOOLS	X					X	
NETWARE 4.0 ELECTRONIC DOCUMENTATION	X			X	X		

X = WORKSTATION HAS THE CAPABILITY TO ACCESS THE NETWARE FEATURE FROM THE SESSION TYPE.
P = PARTIAL ACCESS CAPABILITY.

Table 1: Netware session support for OS2 2.1. The greatest number of supported options are provided for 1) OS/2 sessions and 2) private sessions using a real DOS kernel and running the NetWare 4.0 DOS workstation VLMs. They are the only session types which have NetWare Directory Services (NDS) support. However, all types of sessions do have NetWare 3.11 bindery support. Partial print management for DOS sessions means that while NetWare 4.0 print utilities may be used to manage print jobs, only queues which are in the bindery context of NetWare 3.11 server objects may be viewed.

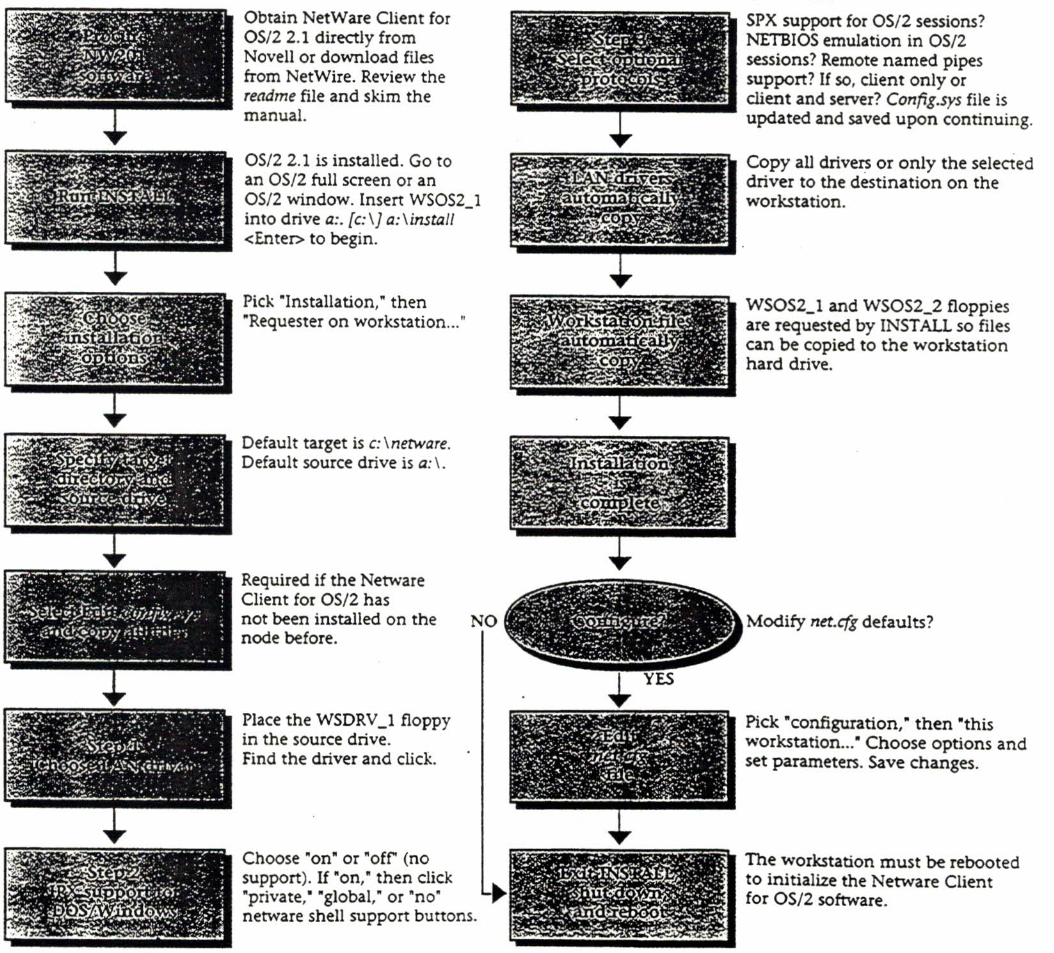
NetWare 3.X server and client drivers. Compatibility of the new frame type with existing networks must be assessed. Some routers do not support the Ethernet 802.2 frame type for IPX. Workarounds deploying frame types other than the new default are outlined. Login from a global DOS session is not supported; Novell recommends login from an OS/2 session if the configuration specifies global DOS and Windows IPX support.

For login to a newly installed (rather than upgraded) NetWare 4.X server from a private DOS session via NETX, there are not any login scripts in place. However, a login script may be created on the workstation's local drive and run with LOGIN /S before the script's path and file name. A private DOS session booted from a real DOS kernel running the NetWare 4.0 DOS workstation VLMs is susceptible to errors from some of the utility programs; PCONSOLE, NETUSER, FILER, NDIR, and NPRINT are vulnerable. Login to the same file server from an OS/2 session helps to remedy the problem. The readme file goes on to point out an error in the NetWare for OS/2 manual for the REQUEST RETRIES setting under NETWARE REQUESTER in the NET.CFG file. The default setting should be 10 rather than 20 retries.

Launching Install

After arming ourselves with knowledge from the readme file and taking the manual in hand, we were ready to begin the installation. The install program on the WSOS2_1 floppy may be run from an OS/2 Window or full screen. After installing all versions of the NetWare Requester for OS/2 since inception, it was a pleasure to see how far the Presentation Manager based INSTALL.EXE has developed in terms of user friendliness and features. The NetWare Workstation for OS/2 Installation Utility screen appears with an inviting greeting and straightforward instruction bullets. All key installation options are on the "Installation" menu.

Table 2: First-time installation flow diagram for Netware Client 2.01 for OS/2.



Events	Comments
Procure NW2.01 software	Obtain NetWare Client for OS/2 2.1 directly from Novell or download files from NetWire. Review the *readme* file and skim the manual.
Run INSTALL	OS/2 2.1 is installed. Go to an OS/2 full screen or an OS/2 window. Insert WSOS2_1 into drive *a:*. *[c:\] a:\install* <Enter> to begin.
Choose installation options	Pick "Installation," then "Requester on workstation..."
Specify target directory and source drive	Default target is *c:\netware*. Default source drive is *a:\*.
Select Edit *config.sys* and copy utilities	Required if the Netware Client for OS/2 has not been installed on the node before.
Step 1 Choose LAN driver	Place the WSDRV_1 floppy in the source drive. Find the driver and click.
Step 2 IPX support for DOS/Windows	Choose "on" or "off" (no support). If "on," then click "private," "global," or "no" netware shell support buttons.
Step 3 Select optional protocols	SPX support for OS/2 sessions? NETBIOS emulation in OS/2 sessions? Remote named pipes support? If so, client only or client and server? *Config.sys* file is updated and saved upon continuing.
LAN drivers automatically copy	Copy all drivers or only the selected driver to the destination on the workstation.
Workstation files automatically copy	WSOS2_1 and WSOS2_2 floppies are requested by INSTALL so files can be copied to the workstation hard drive.
Installation is complete	
Configure?	Modify *net.cfg* defaults?
Edit *net.cfg* file	Pick "configuration," then "this workstation..." Choose options and set parameters. Save changes.
Exit INSTALL shut down and reboot	The workstation must be rebooted to initialize the Netware Client for OS/2 software.

Below "Installation", we chose "Requester on workstation...". From there, we had to confirm the source drive, A:, and destination directory on the OS/2 workstation's hard disk, C:\NETWARE, to where the files would be copied. Next, we were faced with choices of whether or not to edit the CONFIG.SYS file and which files we wanted to copy. Options are provided to: edit CONFIG.SYS and copy all files, only edit the CONFIG.SYS file, only copy the requester files, and only copy ODI LAN driver files. We relied on the help screens to understand when each option should be used. Since the NetWare Workstation for OS/2, version 2.01 was not yet installed on the workstation, our choice was to edit CONFIG.SYS and copy all files. The other three options are designed for partial installations or CONFIG.SYS editing that might be required down the road.

The "edit CONFIG.SYS and copy all files" option has three steps. Step 1 requires selection of a default LAN driver. The WSDRV-1 floppy must be placed in the source drive so the installer can review the drivers and click on a selection. We chose TOKEN.SYS for IBM Token-Ring. Step 2 allows NetWare support for DOS and Windows applications to be turned on (the default is "off"). We picked the "on" button so we could fully experience OS/2 as a "better Windows than Windows." In addition, we clicked "Global NetWare shell support" instead of "Private NetWare shell support" since we wanted the same logins and drive mappings across all OS/2 and DOS sessions. With "Private" sessions, logins and drive mappings would be unique for each session. Step 3 involves optional protocols for OS/2 sessions. We selected SPX support, NetBIOS emulation, and remote named pipes support (client only). After completing the optional protocol selections, the installation program updated the CONFIG.SYS file.

Before the files actually copy to the workstation hard disk, INSTALL asks whether the default driver or all drivers should be copied. We requested all drivers in order to have them readily available for perusal and experimentation with different NICs and various configurations. The ODI LAN drivers and requester files were then copied to C:\NETWARE from WSDRV_1. WSOS2_1 and WSOS2_2 had to be inserted into the source drive to complete copying the full file set to C:\NETWARE. (Also, files for NetWare WIN_OS/2 support, such as NWIPXSPX.DLL, NETWARE.DRV, NETAPI.DLL, and TBMI2.COM, were automatically copied to C:\OS2\MDOS\WINOS2\SYSTEM.) At that point, installation of the software was finished and the NetWare Workstation for OS/2 could initialize upon rebooting. The INSTALL program informed us that we might need or want to configure the workstation. That means edit the NET.CFG file via the "Configuration" menu option and "This workstation..." below it.

Configuration is exercised for any number of reasons. INSTALL suggests we might find it desirable to:
1) define a "NetWare Requester Name Context" statement to avoid typing in our context on the command line when we login,
2) include both Ethernet 802.3 frame type as well as 802.2, the new default, for backward compatibility with older NetWare 3.X and 2.X drivers running in production Ethernet,
3) use a NetWare Requester Preferred Tree statement if there is more than one directory tree,
4) deploy a Link Driver statement if the NIC is not set to default settings,
5) define a Link Driver statement for each NIC if there are multiple NICs in the workstation,
6) utilize statements allowing the NetWare Workstation to share a NIC with other communication software, such as IBM Extended Services, IBM LAN Services, or Novell's NetWare for OS/2, and
7) configure parameters for special "resource settings" pertinent to IPX, SPX, Named Pipes, Novell NetBIOS, Token-Ring source routing, maximum packet size, and so on. We will look at NET.CFG settings in the configuration section.

CONFIG.SYS

The INSTALL program modified path lines in the OS/2, v2.1 *CONFIG.SYS* file, changing them to what are shown below.

```
LIBPATH=.
;C:\OS2\DLL;C:\OS2\MDOS;
C:\;C:\OS2\APPS\DLL;
```

```
C:\MMOS2\DLL;C:\NETWARE;

SET PATH=C:\OS2;
C:\OS2\SYSTEM;
C:\OS2\MDOS\WINOS2;
C:\OS2\ INSTALL;
C:\;C:\OS2\MDOS;
C:\OS2\APPS;C:\MMOS2;
C:\NETWARE;L:\OS2; P:\OS2;

SET DPATH=
C:\OS2;C:\OS2\SYSTEM;
C:\OS2\MDOS\WINOS2;
C:\OS2\ INSTALL;C:\;
C:\OS2\BITMAP;C:\OS2\MDOS;
C:\OS2\APPS;C:\MMOS2;
C:\MMOS2\INSTALL;
C:\NETWARE;L:\OS2;P:\OS2;
```

Directories C:*\NETWARE*, L:\OS2, and P:\OS2 were added. Since NetWare for OS/2 workstations do not have search drives, like their NetWare Workstation for DOS siblings, all searching must be done through the *LIBPATH*, *PATH*, and *DPATH*. Remember C:*\NETWARE* is the subdirectory to where the NetWare workstation drivers and "requester" files were copied.

L:\OS2 corresponds to a file server attachment mapping. Upon attaching to a file server after the OS/2 workstation initializes, L: is actually map rooted to the file server *LOGIN* subdirectory. LOGIN contains only DOS programs. The workstation must be able to search *LOGIN/OS2* for the *OS/2 LOGIN.EXE* and related LOGIN programs. Likewise, P: will be map rooted to the file server's PUBLIC subdirectory after login. Using the same logic, the OS/2 PUBLIC programs reside in *PUBLIC\OS2*.

In addition to the modified path lines, the following entries were appended to the OS/2 *CONFIG.SYS*. They initialize the NetWare Workstation for OS/2 software via drivers and daemons based on configuration choices made in the *INSTALL* program.

```
REM — NetWare Requester statements BEGIN —
SET NWLANGUAGE=ENGLISH
DEVICE=C:\NETWARE\LSL.SYS
RUN=C:\NETWARE\DDAEMON.EXE
DEVICE=C:\NETWARE\TOKEN.SYS
DEVICE=C:\NETWARE\ROUTE.SYS
DEVICE=C:\NETWARE\IPX.SYS
DEVICE=C:\NETWARE\SPX.SYS
RUN=C:\NETWARE\SPDAEMON.EXE
DEVICE=C:\NETWARE\NMPIPE.SYS
rem DEVICE=C:\NETWARE\NPSERVER.SYS
RUN=C:\NETWARE\NPDAEMON.EXE
DEVICE=C:\NETWARE\NWREQ.SYS
IFS=C:\NETWARE\NWIFS.IFS
RUN=C:\NETWARE\NWDAEMON.EXE
DEVICE=C:\NETWARE\NETBIOS.SYS
RUN=C:\NETWARE\NBDAEMON.EXE
DEVICE=C:\NETWARE\VIPX.SYS
DEVICE=C:\NETWARE\VSHELL.SYS GLOBAL
REM — NetWare Requester statements END —
```

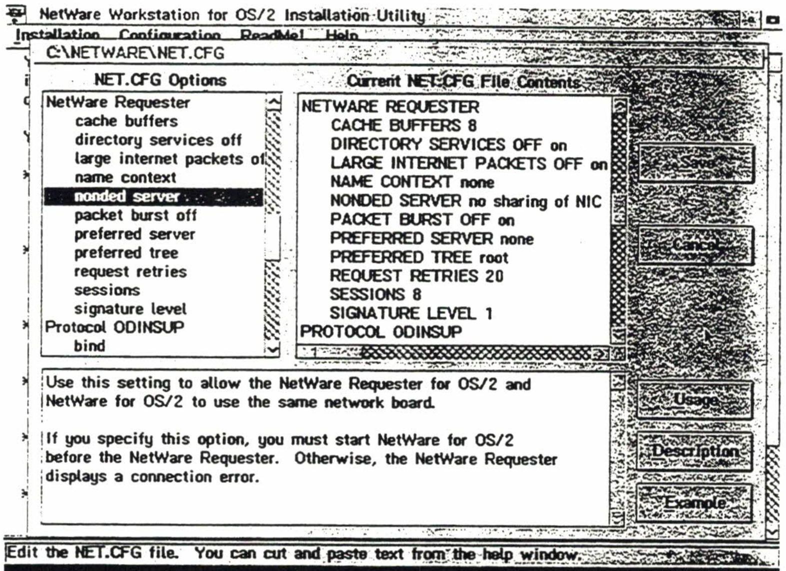
After the language is set to English, the NetWare section of the CONFIG.SYS builds an ODI structure. *LSL.SYS* and *DDAEMON.EXE* load at the link support layer. The Token-Ring LAN driver, *TOKEN.SYS* provides the link layer interface to the IBM Token-Ring adapter, to cement together the physical and link layers. Its companion, *ROUTE.SYS*, facilitates IBM Token-Ring source routing. Next, the protocols, i.e., IPX, SPX, Named Pipes, and (later) NetBIOS load with their respective daemons. SPX and NetBIOS sit on IPX while Named Pipes rides SPX.

If the *NPSERVER.SYS* line was unremarked and a name placed after the *NPDAEMON.EXE* line, the node would initialize as a SAPing Named Pipes server daemon. The server daemon can support Microsoft SQL Server or another backend application platform designed to utilize the Named Pipes protocol. *NWREQ.SYS* is the actual workstation or "requester". *NWIFS.IFS* initializes the NetWare Installable File System. *VIPX.SYS* and *VSHELL.SYS* are enabled for global virtual DOS/Windows sessions. Private virtual DOS/Windows sessions require only *VIPX.SYS*; afterward, NETX must be loaded manually or via the *AUTOEXEC.BAT* file. Physical, link, network, transport, and session layers have been initialized by the end of the *CONFIG.SYS*.

INSTALL ad nauseum

There are other noteworthy options under "Installation" on the main menu. Below "Requester on workstation..." is "NSD on workstation..." for applying NSDs. While we optimistically hope there will not be any (or many) NSDs this time around, the feature is a nice touch. Options for installing and configuring remote workstations (that is, workstations without hard disks) and for putting the NetWare for OS/2 driver and file server files onto the workstation are respectively the third and fourth options listed under "Installation". The NetWare for OS/2 (file server) files are copied from supplemental floppies that must be separately obtained from Novell.

The "Configuration" option brings up two selections whereby the *NET.CFG* for "This workstation..." or "Remote workstations..." may be edited. The editor brings up three display panels. The upper rightmost panel displays the current NET.CFG file. All possible *NET.CFG* options are listed next to it in the upper left panel. When the cursor bar highlights each option, information about the option is displayed in the lower panel. Information in the lower panel may be switched between "Description", parameter "Usage", and "Example". Editing and referencing may be done at the same time in a very efficient manner. A third main item displays the *README.1ST* file that we looked at before launching INSTALL. The fourth main menu option provides help screens.


NetWare Workstation for OS/2 Installation Utility
Installation
Configuration
ReadMe!
Help
C:\NETWARE\NET.CFG
NET.CFG Options
Current NET.CFG File Contents
NetWare Requester
cache buffers
directory services off
large internet packets of
name context
nonded server
packet burst off
preferred server
preferred tree
request retries
sessions
signature level
Protocol ODINSUP
bind
Save
Cancel
Usage
Description
Example

```
NETWARE REQUESTER
    CACHE BUFFERS 8
    DIRECTORY SERVICES OFF on
    LARGE INTERNET PACKETS OFF on
    NAME CONTEXT none
    NONDED SERVER no sharing of NIC
    PACKET BURST OFF on
    PREFERRED SERVER none
    PREFERRED TREE root
    REQUEST RETRIES 20
    SESSIONS 8
    SIGNATURE LEVEL 1
PROTOCOL ODINSUP
```

Use this setting to allow the NetWare Requester for OS/2 and NetWare for OS/2 to use the same network board.

If you specify this option, you must start NetWare for OS/2 before the NetWare Requester. Otherwise, the NetWare Requester displays a connection error.

Edit the NET.CFG file. You can cut and paste text from the help window.

Figure 5: Configuration and This Workstation lead to editing the *net.cfg* file and referencing *net.cfg* parms. This facility is new for version 2.0x. Usage, Description and Example buttons alter the lower panel information text for each option line.

Documentation of the NetWare Workstation for OS/2 is extensive at 300 plus pages. The combination of IBM OS/2, version 2.1 and Novell NetWare Workstation for OS/2, version 2.01 is a "powerful" offering that demands a good manual.

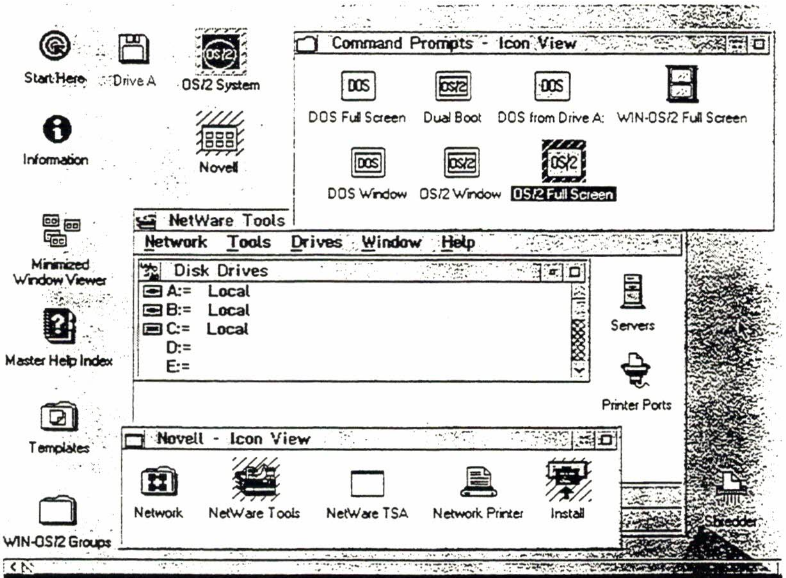

Start Here
Drive A
OS/2 System
Information
Novell
Command Prompts - Icon View
DOS Full Screen
Dual Boot
DOS from Drive A:
WIN-OS/2 Full Screen
DOS Window
OS/2 Window
OS/2 Full Screen
NetWare Tools
Network
Tools
Drives
Window
Help
Disk Drives
A:= Local
B:= Local
C:= Local
D:=
E:=
Servers
Printer Ports
Minimized Window Viewer
Master Help Index
Templates
Novell - Icon View
Network
NetWare Tools
NetWare TSA
Network Printer
Install
WIN-OS/2 Groups
Shredder

Figure 6: The Novell Icon View sits proudly on the OS/2 desktop. The familiar NetWare Tools is especially useful. A network folder lets us select detailed file-server resources across the enterprise network. NetWare support capabilities vary across different kinds of sessions.

The manual comes in the $30 package from Novell or may be downloaded from NetWire in the form of 17 zipped postscript files. After downloading, the files can be displayed on a postscript printer. The documentation is detailed, well written, easy to understand, and chock full of juicy configuration details. The days are long gone when the local Novell branch office had to fax supplemental undocumented material in order for our configuration gymnastics to be completed. Graphic illustrations reveal the underlying interoperability combinations to facilitate full IBM-Novell integration. IBM must be quite pleased with this Novell manual.

HARMONY

Upon rebooting, installation and configuration confusion turns into operational harmony. A Novell icon and a network folder appear on the desktop after the software comes up. Within Novell, there are various icons which represent Install, NetWork Printer (NPrinter), NetWare Storage Management System (SMS) Target Service Agent (TSA) for OS/2, and NetWare Tools. NPrinter may be used to connect or disconnect to a remote printer on the network. NetWare SMS TSA facilitates SBACKUP of the workstation's hard drive. NetWare Tools is a multifunction utility for attachment, login, mapping drives, messaging, print capturing, and so on. The network folder displays an icon for every NetWare server on the network. A click on a server icon brings up icons and folders that are pertinent to print queues and file server volumes.

But curiosity beckons us back to the configuration process. Should we stick with a "straight" *TOKEN.SYS* or Ethernet driver, or bring *ODINSUP.SYS* or *LANSUP.SYS* into the picture along with the Communications Manager of IBM Extended Services? Why not also throw IBM LAN Services into the mix? We could deploy Novell NetBIOS and IBM NetBIOS concurrently. Where is TCP/IP? ISO? How about creating some individual DOS/Windows sessions with real DOS kernels or private virtual DOS? And let's not forget the NetWare for OS/2 nondedicated file server that we yearn to get our hands on. For now, we will resist the enticement. Enough is enough lest we fly too close to the sun.

In conclusion, we were very pleased to experience the unity of OS/2, version 2.1 and NetWare Workstation for OS/2, version 2.01. A fast 80486 computer with 16 megabytes of RAM and a few hundred megabytes of disk storage is desirable for running them. The joint implementation represents a definite high point in the IBM-Novell alliance. Based on our experience, market impact of the two products working together is bound to be significant. IBM's OS/2 platform has finally matured and established deep roots after six years since the initial release. Indeed, the Caesar of Windows NT should beware the likes of OS/2.

MNC 0310

How to Get the Most Out of Your OS/2 Warp

David Moskowitz
Productivity Solutions
Norristown, Penn.

How to get the most out of your OS/2 Warp

David Moskowitz
Productivity Solutions, Inc.

Agenda

- A quick work about tuning
- Installation
- Tuning
- Command line goodies
- Workplace Shell Stuff

Tuning preamble

- Tuning is monitoring!
- There is no single best setup that is guaranteed to work for everyone.
- 7 systems for 7 people 7 times.

Installation

- Install components you'll use
  - Consider tradeoffs
  - You can always install it later
- Use System configuration menu
  - Move swappath
  - Change minimum space
  - Use the third parameter on SWAPPATH
    - Edit CONFIG.SYS after installation
    - Make it "big enough"

Using large disks

- Some BIOS already handle big IDE drives
- Bootable partitions totally contained in first 528M (1024 cylinders)
- HPFS for the rest
- Ontrack
  - Setup Ontrack partition in DOS (use dynamic mode)
  - Select Advanced install in OS/2 and remove it with OS/2 FDISK and install

System setup & tuning

- Config.sys parameters
- Disk caching
- Swappath location
- DOS settings
- Under the covers information helps

Config.sys parameters

- Don't change THREADS unless you must.
- Try various settings for MAXWAIT
- PRIORITY_DISK_IO for BBS or server
- Set PROTECTONLY to yes if no DOS
- Sometimes RMSIZE=639 helps
- Avoid DOS device drivers in CONFIG.SYS
- SESSION_PRIORITY helps
- Hidden "fixes"
  - BASEDEV=IBM1S506.ADD /U:0 /!SMS
  - IFS=CDFS.IFS /Q /C:4

Disk caching

- What and Why
  - In memory buffer that speed disk access
- Determine size
  - Based upon available RAM and use
  - Which file systems installed.
  - Which one is most frequently used.
- Lazy write
  - Always leave it ON
    - Buy an UPS
- HPFS control -- More flexible than FAT

Determing cache size

- How many file systems?
- Which one is primary?
- How much RAM is installed?
- Determining cache size is a balancing act between improving disk access and increasing the need to use the swap file.
  - Decrease swapping by adding memory.
  - Decrease caching by using a caching controller.

Swappath location

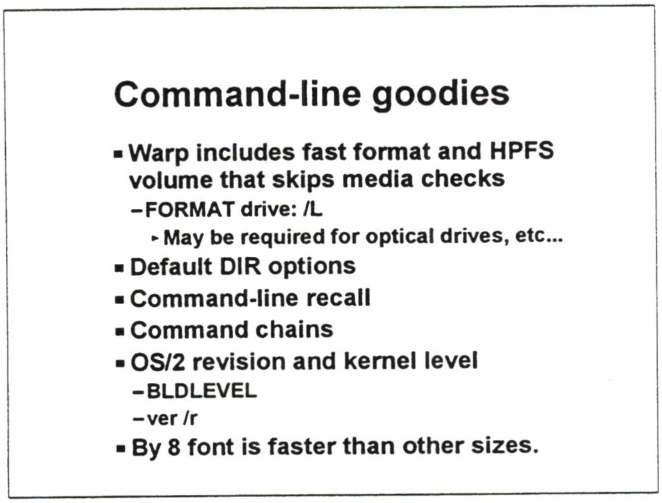
- Move it from \OS2\SYSTEM
- Root directory
- Least frequently used spindle
- Most frequently used partition near edge
- Do not setup separate partition
- Consider a dedicated physical drive
- Avoid fragmentation, start with larger size

DOS Settings

- Separate AUTOEXEC per session
  - DOS_AUTOEXEC
- Background execution
  - DOS_BACKGROUND_EXECUTION
  - INT_DURING_IO
- Special DOS version
  - Doesn't have to be DOS!
  - VMDISK (special startup files)
  - DOS_STARTUP_DRIVE

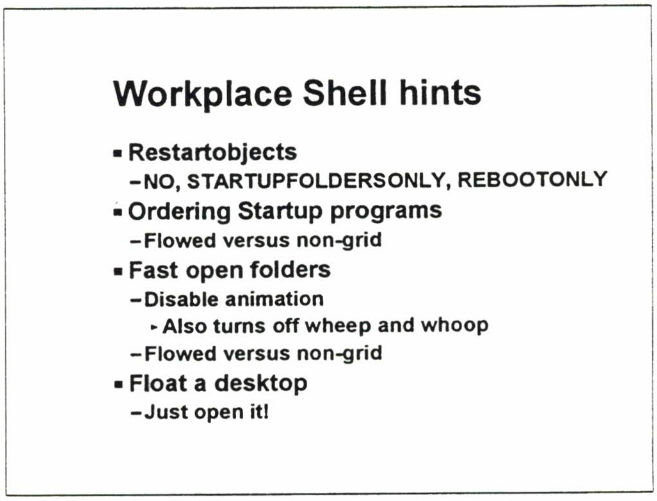
DOS Hints

- Change CONFIG.SYS settings
  - DOS_UMB, DOS_HIGH, etc.
- Toggle window to full screen
  - Alt-Home
- Separate devices
  - Load when necessary in SETTINGS not part of CONFIG.SYS
- Separate environments



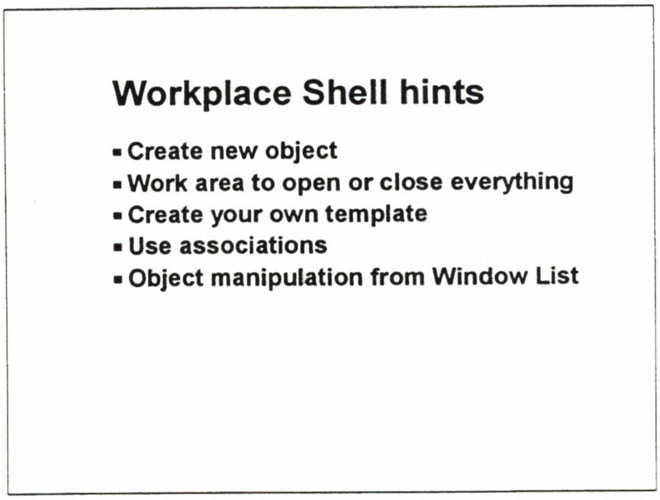
Command-line goodies

- Warp includes fast format and HPFS volume that skips media checks
  - FORMAT drive: /L
    - May be required for optical drives, etc...
- Default DIR options
- Command-line recall
- Command chains
- OS/2 revision and kernel level
  - BLDLEVEL
  - ver /r
- By 8 font is faster than other sizes.



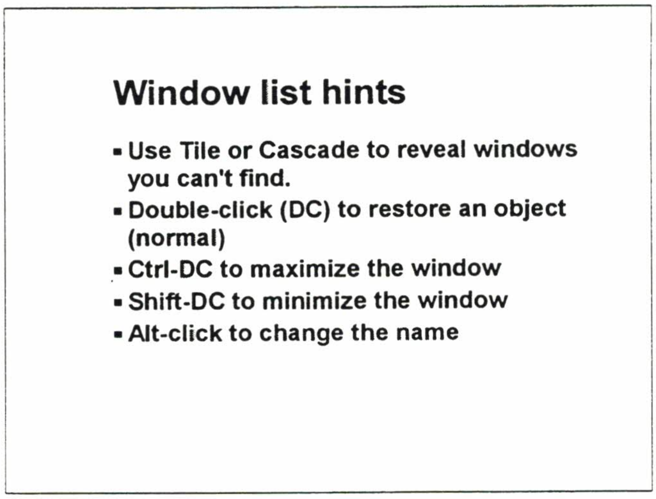
Workplace Shell hints

- Restartobjects
  - NO, STARTUPFOLDERSONLY, REBOOTONLY
- Ordering Startup programs
  - Flowed versus non-grid
- Fast open folders
  - Disable animation
    - Also turns off wheep and whoop
  - Flowed versus non-grid
- Float a desktop
  - Just open it!



Workplace Shell hints

- Create new object
- Work area to open or close everything
- Create your own template
- Use associations
- Object manipulation from Window List



Window list hints

- Use Tile or Cascade to reveal windows you can't find.
- Double-click (DC) to restore an object (normal)
- Ctrl-DC to maximize the window
- Shift-DC to minimize the window
- Alt-click to change the name

LaunchPad goodies

- Double click the desktop to open it (even if closed)
- Tear off drawers
- Orientation
- Settings and shadows
- Launchpad icons are shadows stored in the Nowhere1 folder. Current buglet - doesn't always delete the icon if you remove it from the LaunchPad

Some neat tricks

- Multiple desktop views
- Fast scroll text window
- Save text window size & position
- Change default lines in a text window
- Moving icons
- Scheme palette help
- Move background window
- Undo the icon
- Reboot from the command line

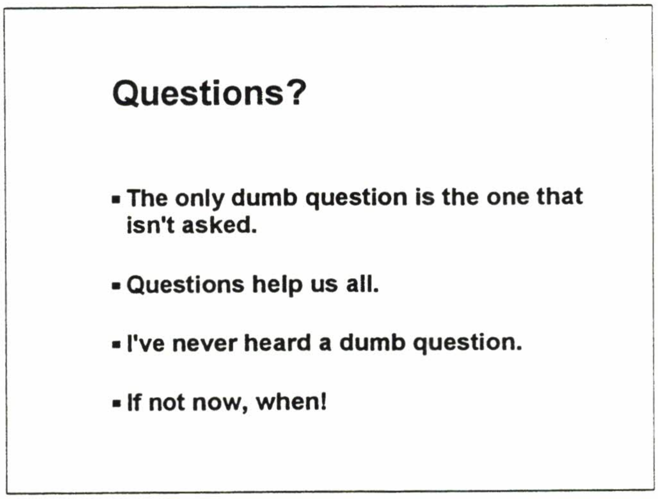
Third party tools

- 4OS2 & 4DOS
- Boot2x
- Deskman/2
- INIMaint et al
- PMMPEG
- SIO drivers
- Partition Magic
- TShell & MShell
- Info ZIP
- Watch Cat & Watch Dog
- Taskbar
- WPSBackup
- GammaTech & Grahmn
- NetSuite
- DInfo
- Memsize
- Back Again/2
- AMOS

Hardware upgrades

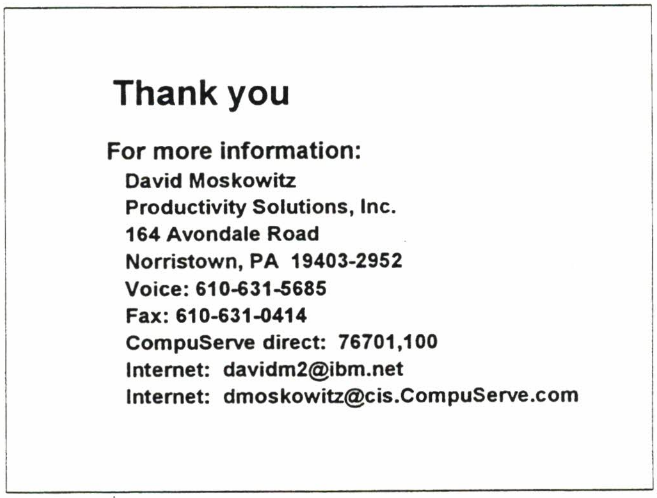
To improve performance of...

- Display -- add a (faster) video accelerator
  - Note: if your video is slow you won't see much difference with the rest
  - Match monitor to adapter
  - Make sure combination is flicker free
- Applications -- add memory
- Disk speed -- add hardware cache
  - Match SCSI devices to adapter
- Computation -- upgrade the processor



Questions?

- The only dumb question is the one that isn't asked.
- Questions help us all.
- I've never heard a dumb question.
- If not now, when!



Thank you

For more information:

David Moskowitz
Productivity Solutions, Inc.
164 Avondale Road
Norristown, PA 19403-2952
Voice: 610-631-5685
Fax: 610-631-0414
CompuServe direct: 76701,100
Internet: davidm2@ibm.net
Internet: dmoskowitz@cis.CompuServe.com

this page intentionally left blank

PRO 0304

Introduction to REXX part 2

Chip Davis
Hilltop, Maryland

INTRODUCTION

Welcome back. In "Intro to Rexx - Part 1" we used the "Flip-a-coin Executive Decision Maker" to introduce some fundamental concepts of Rexx. In "Part 2", we will continue to explore Rexx by showing you how to write your own subroutines and functions, perform I/O, use compound variables, parse strings, issue external commands and process the responses, and do a little bit of debugging as we go along. That's a lot of material, so let's get started.

SUBROUTINES

When we last left our hero, rexx7.cmd had discovered how to escape from the evil clutches of an infinite loop by

```
<Listing 8>

/* rexx8.cmd: Flip-a-coin Executive Decision Maker */ /*  1 */
  heads = 1                                           /*  2 */
  tails = 2                                           /*  3 */
  Call Help                                           /*  4 */
  Do num = 1                                          /*  5 */
    Say “”                                            /*  6 */
    Say “What is question number” num”?”              /*  7 */
    Pull quest                                        /*  8 */
    If quest = ‘QUIT’ Then Leave num                  /*  9 */
    If quest = ‘HELP’ Then Call Help                  /* 10 */
    If Right(quest,1) \= ‘?’ Then Do                  /* 11 */
      Say “That is not a question.”                   /* 12 */
      Iterate num                                     /* 13 */
    End                                               /* 14 */
    answer = Random(1,3)                              /* 15 */
    Select                                            /* 16 */
      When answer = heads Then Say “Yes.”             /* 17 */
      When answer = tails Then Say “No.”              /* 18 */
      Otherwise Say “Maybe.”                          /* 19 */
    End                                               /* 20 */
  End num                                             /* 21 */
  Say “”                                              /* 22 */
  Say “Goodbye.”                                      /* 23 */
  Return                                              /* 24 */
/* Help subroutine */                                 /* 25 */
Help:                                                 /* 26 */
  Say “The forces that shape our lives are complex,”  /* 27 */
  Say “but recent advances in decision-making by the” /* 28 */
  Say “Congressional Budget Office promise to make”   /* 29 */
  Say “dealing with tough choices much simpler.  By”  /* 30 */
  Say “using a specially-designed, three-sided Susan” /* 31 */
  Say “B. Anthony dollar (heavily weighted on the”    /* 32 */
  Say “right side), Congress was able to complete its”/* 33 */
  Say “version of the Federal Budget in record time.” /* 34 */
  Say “Now you too can make those tough decisions”    /* 35 */
  Say “without the bother of actually having to think”/* 36 */
  Say “about the problem.  Simply ask your yes-or-no” /* 37 */
  Say “question, and your personal digital lobbyist”  /* 38 */
  Say “will tell you what you think.  Enter as many”  /* 39 */
  Say “yes/no questions as you like.  To repeat this” /* 40 */
  Say “help message, enter ‘help’.  To leave before”  /* 41 */
  Say “the special prosecutor is appointed, enter”    /* 42 */
  Say “‘quit’.”                                       /* 43 */
  Return                                              /* 44 */
```

uttering the somewhat obvious incantation "Leave". We now have a minimal, but functional program. One thing that is still missing is any sort of user help besides the prompt. User help can be placed inside or outside a program. The advantage of having it external to the program is that the program is much smaller. The disadvantage is that it's hard enough to keep the documentation in synch with changes to the program when it's imbedded in the code. It's a herculean effort when they are separate files. Despite the fact that it will almost double the size of our program, we're going to include a modest help paragraph in rexx8.cmd. The problem is that we want to provide this help at two points in the program: at the introduction, and whenever the user asks for it.

This calls for a SUBROUTINE. A subroutine is a section of code that may be invoked as a unit from somewhere else in the program. Subroutines begin with a LABEL, which marks the entry point and provides a unique name for the routine (line 26 of Listing 8). A label looks like a variable name with a colon (:) appended. You invoke the subroutine by name, using the CALL keyword instruction. 'Call' is followed by the name of the subroutine (don't include the colon) followed by any arguments you may wish to pass to the routine (lines 4 and 10). Multiple arguments are separated by commas (,). When Rexx encounters a subroutine call, it saves its place and transfers control to the subroutine.

When the subroutine is through, it uses a RETURN keyword instruction (line 44) to return control back to the instruction following the 'Call', and execution continues. Yes, this is the same 'Return' instruction that you use to terminate your Rexx program, which is really only a subroutine of some other process, which is really a subroutine of yet another process ... No one codes on bare metal anymore.

Let's try this out (Screen 8).

```
<Screen 8>

[D:\]rexx8
The forces that shape our lives are complex,
but recent advances in decision-making by the
Congressional Budget Office promise to make
dealing with tough choices much simpler. By
using a specially-designed, three-sided Susan
B. Anthony dollar (heavily weighted on the
right side), Congress was able to complete its
version of the Federal Budget in record time.
Now you too can make those tough decisions
without the bother of actually having to think
about the problem. Simply ask your yes-or-no
question, and your personal digital lobbyist
will tell you what you think. Enter as many
yes/no questions as you like. To repeat this
help message, enter 'help'. To leave before
the special prosecutor is appointed, enter
'quit'.

What is question number 1?
Wow! Is all that really necessary?
Maybe.

What is question number 2?
quit

Goodbye.
```

FUNCTIONS

Another reason to structure or modularize your program is to get the details out of the way so you can see the underlying logic. You may have a group of instructions that together perform a particular function. Coding the instructions in-line with the rest of your program ensures efficient execution, but at the expense of program readability. In Listing 9, we have taken the instructions that generated the random answer and separated them from the primary logic of the program. This allows us to greatly simplify the main loop by using a 'Select' instead of an assortment of 'If's and Do-groups (lines 10-15). We've created what looks like another subroutine, called 'Answer', which is invoked from one of the 'When' instructions. Actually, "Answer" isn't a subroutine, it's a FUNCTION. Look closely at "Answer" (lines 21-27) and "Help" (lines 29-47). Can you tell the difference between the function and the subroutine? Me neither.

Subroutines and functions are both more properly referred to as SUBPROCEDURES. A subprocedure may be invoked as a subroutine or as a function. You may pass arguments to a subprocedure, and get a value returned from it. There are only two significant distinctions between subroutines and functions: 1) how they are invoked, and 2) where the returned value ends up.

A function is invoked by using its name in an expression. Arguments are separated by commas, enclosed in parenthe-

<Listing 9>

```
/* rexx9.cmd: Flip-a-coin Executive Decision Maker */ /*  1 */
  heads = 1                                            /*  2 */
  tails = 2                                            /*  3 */
  Call Help                                            /*  4 */
  Do num = 1                                           /*  5 */
    Say “”                                             /*  6 */
    Say “What is question number” num”?”               /*  7 */
    Pull quest                                         /*  8 */
    answer = Random(1,3)                               /*  9 */
    Select                                             /* 10 */
      When quest = ‘QUIT’       Then Leave num         /* 11 */
      When quest = ‘HELP’       Then Call Help         /* 12 */
      When Right(quest,1) = ‘?’ Then Say Answer()      /* 13 */
      Otherwise Say “That is not a question.”          /* 14 */
    End /* Select */                                   /* 15 */
  End num                                              /* 16 */
  Say “”                                               /* 17 */
  Say “Goodbye.”                                       /* 18 */
  Return 0                                             /* 19 */
/* Answer the question */                              /* 20 */
Answer:                                                /* 21 */
  Select                                               /* 22 */
    When answer = heads Then ans = “Yes.”              /* 23 */
    When answer = tails Then ans = “No.”               /* 24 */
    Otherwise ans = “Maybe.”                           /* 25 */
  End                                                  /* 26 */
  Return ans                                           /* 27 */
/* Help subroutine */                                  /* 28 */
Help:                                                  /* 29 */
  Say “The forces that shape our lives are complex,”   /* 30 */
  Say “but recent advances in decision-making by the”  /* 31 */
  Say “Congressional Budget Office promise to make”    /* 32 */
  Say “dealing with tough choices much simpler.  By”   /* 33 */
  Say “using a specially-designed, three-sided Susan”  /* 34 */
  Say “B. Anthony dollar (heavily weighted on the”     /* 35 */
  Say “right side), Congress was able to complete its”/* 36 */
  Say “version of the Federal Budget in record time.”  /* 37 */
  Say “Now you too can make those tough decisions”     /* 38 */
  Say “without the bother of actually having to think”/* 39 */
  Say “about the problem.  Simply ask your yes-or-no”  /* 40 */
  Say “question, and your personal digital lobbyist”   /* 41 */
  Say “will tell you what you think.  Enter as many”   /* 42 */
  Say “yes/no questions as you like.  To repeat this”  /* 43 */
  Say “help message, enter ‘help’.  To leave before”   /* 44 */
  Say “the special prosecutor is appointed, enter”     /* 45 */
  Say “‘quit’.”                                        /* 46 */
  Return                                               /* 47 */
```

ses and abutted to the function name. Even if you have no arguments to pass (line 13), the parens are required for Rexx to know that you aren't referring to a simple variable name. The result of executing the function is returned to the expression, where it replaces the function invocation, and expression evaluation continues. The returned value might be used in further operations, assigned to a variable, tested by an 'If', sent to the screen with a 'Say', in short, anything that you might use a variable for.

What happens if the function doesn't return a value? The expression where it was invoked blows up. How do you return a value from a function? Specify it on the 'Return' instruction. Rexx evaluates the expression following the 'Return' keyword, and that value is passed back. While a syntax diagram may indicate that the expression is optional, if the subprocedure was invoked as a function, you jolly-well better return something!

A subroutine, on the other hand, is invoked by using its name in the 'Call' instruction. Arguments are specified following the subroutine name, and are separated from each other by commas. The returned value is assigned to a Rexx SPECIAL VARIABLE. A special variable is just an otherwise ordinary variable that Rexx uses for a special purpose. There are only three of them, and they are not reserved words, nor off-limits to you. The special variable that contains the returned value from a subroutine is RESULT. You may use 'Result' as you would any other Rexx variable; just be aware that you are sharing it with Rexx. If your program doesn't call any subroutines, Rexx won't disturb your value. A subroutine is not required to return a value. If it does, the value goes into 'Result'. If it doesn't, 'Result' goes back to its default value ("RESULT").

```
<Screen 9>

[D:\]rexx9
The forces that shape our lives are complex,
but recent advances in decision-making by the
Congressional Budget Office promise to make
dealing with tough choices much simpler. By
using a specially-designed, three-sided Susan
B. Anthony dollar (heavily weighted on the
right side), Congress was able to complete its
version of the Federal Budget in record time.
Now you too can make those tough decisions
without the bother of actually having to think
about the problem. Simply ask your yes-or-no
question, and your personal digital lobbyist
will tell you what you think. Enter as many
yes/no questions as you like. To repeat this
help message, enter 'help'. To leave before
the special prosecutor is appointed, enter
'quit'.

What is question number 1?
Will this work?
  27 +++        Return;
REX0045: Error 45 running D:\rexx9.cmd, line 27: No data
specified on function RETURN
  13 +++      Say Answer();
```

When should you use a function, and when should you use a subroutine? As you can see, it matters not one whit to the subprocedure itself. In fact, as long as you always 'Return' a value of some sort (even if it's just a zero, or a null string), it doesn't really matter to you either. Generally speaking, you invoke a subprocedure as a function when you want to use the value it returns. You call it as a subroutine when you just want to get some work done. Yes, a concientious programmer will test the value in 'Result' after the 'Call', but it's not required, and often unnecessary. (Certain system routines, eg. 'SysCls' which clears the screen, always return zero.) In Listing 9, we invoked "Answer" as a function, because it would have taken two instructions to get the answer to the screen otherwise ("Call Answer" and "Say Result"). That would have necessitated a Do-group for the 'When', and things would have gotten messy.

These changes caused no discernable difference in the execution of our program (trust me). Instead Screen 9 shows what happened when I removed the "ans" variable from the 'Return' instruction in the "Answer" subprocedure. Rexx pretty much tells you where it hurts.

INPUT/OUTPUT

Rexx was designed to be platform-independent, remember? Can you imagine how difficult it was to design an I/O mechanism that could be implemented on operating systems as diverse as MVS and Unix? Yet the Rexx I/O model is simple and elegant. There are mechanisms for handling all manner of platform-specific I/O functions, but the fundamental operations are very simple: you can read and write data as lines (records), or as characters (or both). It even supports I/O redirection and interprocess communication queues.

In Listing 10, we've decided that it would be better to retrieve the potential answers and the help text from files, rather than hard-coding them in the program. (Of course, we are still limited to only three answers, because of our random selection logic. We'll fix that in a minute.) At least we had the sense to put the file ids into variables (lines 4-5). It's a lot easier to change them later if they are in one place, than to try to find every I/O instruction.

Rexx I/O is performed by the LINEIN, LINEOUT, CHARIN, and CHAROUT routines. CHARS and LINES indicate whether or not there is something left to read. STREAM handles everything else, from explicit open/close, to positioning within the file, to extracting descriptive information.

Let's look at the "Get_Answers" subroutine. The first thing we want to do is verify that the answer file is accessible. This is done by passing to the 'Stream' routine the name of our file as the first argument, followed by the two specific arguments 'Command' and 'Query Exists' (line 25). If Rexx can find the file, 'Stream' returns its full

<Listing 10>

```
/* rexx10.cmd: Flip-a-coin Executive Decision Maker *//*  1 */
  heads = 1                                           /*  2 */
  tails = 2                                           /*  3 */
  ansdata = ‘rexx10.ans’                              /*  4 */
  hlpdata = ‘rexx10.hlp’                              /*  5 */
  Call Get_Answers                                    /*  6 */
  Call Help                                           /*  7 */
  Do num = 1                                          /*  8 */
    Say “”                                            /*  9 */
    Say “What is question number” num”?”              /* 10 */
    Pull quest                                        /* 11 */
    answer = Random(1,3)                              /* 12 */
    Select                                            /* 13 */
      When quest = ‘QUIT’       Then Leave num        /* 14 */
      When quest = ‘HELP’       Then Call Help        /* 15 */
      When Right(quest,1) = ‘?’ Then Say Answer()     /* 16 */
      Otherwise Say “That is not a question.”         /* 17 */
    End /* Select */                                  /* 18 */
  End num                                             /* 19 */
  Say “”                                              /* 20 */
  Say “Goodbye.”                                      /* 21 */
  Return 0                                            /* 22 */
/* Get answers from file */                           /* 23 */
Get_Answers:                                          /* 24 */
  Call Stream ansdata, ‘Command’, ‘Query Exists’      /* 25 */
  If Result \= ‘’ Then Do                             /* 26 */
    Call Stream ansdata, ‘Command’, ‘Open Read’       /* 27 */
    ans1 = LineIn(ansdata)                            /* 28 */
    ans2 = LineIn(ansdata)                            /* 29 */
    ans3 = LineIn(ansdata)                            /* 30 */
    Call Stream ansdata, ‘Command’, ‘Close’           /* 31 */
  End                                                 /* 32 */
  Else Do                                             /* 33 */
    Say “Unable to access” ansdata “file.”            /* 34 */
    Exit 2                                            /* 35 */
  End                                                 /* 36 */
  Return 0                                            /* 37 */
/* Answer the question */                             /* 38 */
Answer:                                               /* 39 */
  Select                                              /* 40 */
    When answer = heads Then ans = ans1               /* 41 */
    When answer = tails Then ans = ans2               /* 42 */
    Otherwise ans = ans3                              /* 43 */
  End                                                 /* 44 */
  Return ans                                          /* 45 */
/* Provide Help */                                    /* 46 */
Help:                                                 /* 47 */
  If Stream(hlpdata, ‘Command’, ‘Query Exists’) \= ‘’ Then Do
    Call Stream hlpdata, ‘Command’, ‘Open Read’       /* 49 */
    Do While Chars(hlpdata) > 0                       /* 50 */
      Call CharOut ‘STDOUT:’, CharIn(hlpdata)         /* 51 */
    End                                               /* 52 */
    Call Stream hlpdata, ‘Command’, ‘Close’           /* 53 */
  End                                                 /* 54 */
  Else Say “Can’t find” hlpdata”. No help available.” /* 55 */
  Return 0                                            /* 56 */
```

filespec, otherwise it returns a null string. That's why we are testing 'Result' for being non-null in line 26; 'Stream' was called as a subroutine. Contrast that with line 48 in the "Help" routine, where the same operation was invoked as a function. Next we tell 'Stream' to explicitly open our file. Usually this is unnecessary in OS/2; if you try to access an unopened file, it is automatically opened for you. By the same token, it usually isn't necessary to explicitly close a file either; when your program terminates, your files will be closed for you. You'll never go wrong by doing it yourself, though.

'LineIn' reads the next record from the file specified. A "current read position" pointer keeps track of which record that is. If there are no more records, 'LineIn' returns a null string. This is not a reliable indicator of end-of-file, however, as there may be imbedded null records in the file.

The 'Lines' routine returns a "1" if there is at least one more record in the file, and a "0" otherwise. It's not possible to determine exactly how many lines there are in an OS/2 file without pre-reading the entire file. 'Chars', on the other hand, always knows exactly how many characters (bytes) of the file remain. The only problem is that 'Chars' counts control characters as well as data.

'LineOut' accepts two arguments: the filespec and the data to be written. The data will be written at the current write position in the file. On OS/2, the "current write position" is the same as the "current read position", unless you haven't done any reads, in which case it is the end-of-file. In other words, if the first thing you do to a file is a read operation, you will get the first record. If the first thing you do to a file is a write operation, you will append to the end of the file. If you read from the file, then write back to it, the write operation will start at the first character after the last one you read. Makes sense to me.

Yes, it's possible to position yourself at an arbitrary point in the file, by using the 'Seek' command of 'Stream', but you must specify the position in terms of characters. You may specify the seek position relative to the current position, or to the start or end of the file.

Technically, Rexx doesn't do I/O to "files", it does it to "streams" of characters. A stream may be a file on a disk somewhere, or it may be an interprocess communication queue, or it could be a hardware device. It matters not to Rexx; they are all simply a sequence of characters, one after the other. The main difference is the dynamic nature of those characters. Files are nice and predictable; you can easily tell if there is another character available to be read. Files are called "persistent streams". A communications port, or the keyboard, does not give any indication when another character is forthcoming, if at all. These are called "transient streams".

Why the distinction? For starters, it doesn't make much sense to 'Seek' to a position in a stream that may or may not have any more characters in it. If you ask 'Chars' to report how many characters are left in the KBD:, it will have a hard time determining when you're going to stop typing. Since there is always the possibility of a character coming in from a device, 'Chars' and 'Lines' always report "1" for transient streams.

Now look at the 'CharOut' on line 51. The stream in this case is STDOUT:, or the standard output device. Normally it goes to the screen, but there are OS/2 command line operators that allow you to redirect STDIN: (keyboard), STDOUT: (screen), and STDERR: (error messages) elsewhere (to a file maybe, another device, or a program). Assuming no redirection has occurred, this 'CharOut' will send one character at a time to the screen. That's why we need a Do-While loop (lines 50-52). Where is 'CharOut' getting the character each time? From the help data file, via a 'CharIn'.

Before we leave Listing 10, there are two non-I/O items I'd like to point out. In our "Help" routine, if we couldn't access the help data file, we simply put out a message and returned. While an annoyance, not having help is not a big deal. It's a different ballgame if we can't find our answers file, though. What would have happened in "Get_Answers" if we had simply 'Return'ed after writing out our error message in line 34? Exactly; Rexx would have returned to line 7 and tried to run anyway.

The "proper", structured programming approach to this situation would be to return a non-zero value from "Get_Answers", and to test the value of 'Result' after the "Call Get_Answers". If 'Result' is non-zero, a 'Return' at that point would terminate the program, and return control to OS/2, or whomever invoked us. In the real world though, you'll likely discover the problem in a subroutine nested fourteen levels deep. Every intervening level would then have to be smart enough to propagate your return code back up to the top. Ain't likely.

That's why Rexx has EXIT. It is a variant of the 'Return' instruction, identical in every way except that it pretends that it's running at the top level and returns control (and any specified value) to whomever invoked our program. Its only purpose in life is to allow you to terminate your entire program from within a subprocedure. I

```
<Screen 10>

[D:\]type rexx10.ans
Yes.
No.
Maybe.

[D:\]type rexx10.hlp
The forces that shape our lives are complex,
but recent advances in decision-making by the
Congressional Budget Office promise to make
dealing with tough choices much simpler. By
using a specially-designed, three-sided Susan
B. Anthony dollar (heavily weighted on the
right side), Congress was able to complete its
version of the Federal Budget in record time.
Now you too can make those tough decisions
without the bother of actually having to think
about the problem. Simply ask your yes-or-no
question, and your personal digital lobbyist
will tell you what you think. Enter as many
yes/no questions as you like. To repeat this
help message, enter 'help'. To leave before
the special prosecutor is appointed, enter
'quit'.
```

prefer to use the 'Return' instruction for everything except that one special case, because it helps reinforce proper program structure.

The other non-I/O item I'd like to point out is that we have made a marginal improvement in our program by placing the answers in an external file and reading them into variables. It would be fairly easy now to change the "Yes." response to "You bet.", for example, without having to change our program. Unfortunately, the rest of our logic has us locked into having exactly three answers. What would it take to allow for, say, twenty answers? For starters, seventeen more 'LineIn's and seventeen more 'When's. And don't forget to change the upper bound on the 'Random' function, or you'll never see any but the first three answers.

Our next program will show us how to get around all these limitations by using compound variables (and a few other goodies).

The contents of rexx10.ans and rexx10.hlp are displayed in Screen 10.

COMPOUND VARIABLES

Rexx has no data structures, per se. Data structures, like data typing, usually just get in the way of solving a problem. What Rexx provides is an elegantly simple mechanism that allows you to build whatever data structure you like. This is done using COMPOUND VARIABLES. Simple variables have a fixed name, but their value may change during the execution of the program. A compound variable is a meta-variable whose name (as well as value) may change during execution. This allows Rexx to dynamically determine the variables it uses at execution time.

A compound variable name consists of a STEM, or fixed initial part of the variable, followed by one or more ELEMENTS, separated by periods. An element may be a constant or a simple variable. When Rexx encounters a compound variable, it replaces each element with its value. The variable name that results, called the DERIVED NAME, is the one Rexx will then use to look up its value.

The simplest way to demonstrate this is to build an array. Let's say we have the need for a list of, say, twenty responses to a yes/no question. We could easily build such a list by using a sequence of simple variables ("ans1", "ans2", et seqq.), as rexx10.cmd did. The problem comes when we need to refer to a particular variable in the list. Using compound variables, the stem portion ("ans") can be joined to a variable part ("i"), whose value can change during execution to be "1", or "20", or any value we desire. We can then refer to the value stored in the fifth variable by using the compound variable "ans.5", or "ans.rhino" as long as the simple variable "rhino" has the value "5".

That gets us a one-dimensional array or list. Q: What's the maximum size of a Rexx array? A: How much storage have you got? If you've got the space to put the data, Rexx will let you build an array as big as you like. Best of all, unlike most other languages, you don't have to tell Rexx in advance how big it will be.

What if you need a matrix? Use a compound variable with a stem and two elements such as "table.x.y". Three-dimensional space can be represented by a variable like "model.long.wide.deep". How many dimensions do you need to solve your problem? The only limit on the number of dimensions you can represent with a compound

<Listing 11>

```
/* rexx11.cmd: Executive Decision Maker */              /*  1 */
  file = ‘rexx11.ans’                                   /*  2 */
  Call Get_Answers                                      /*  3 */
  Call Help                                             /*  4 */
  Do num = 1                                            /*  5 */
    Say “”                                              /*  6 */
    Say “What is question number” num“?”                /*  7 */
    Pull quest                                          /*  8 */
    answer = Random(1,ans.0)                            /*  9 */
    Select                                              /* 10 */
      When quest = ‘QUIT’       Then Leave num          /* 11 */
      When quest = ‘HELP’       Then Call Help          /* 12 */
      When Right(quest,1) = ‘?’ Then Say ans.answer     /* 13 */
      Otherwise Say “That is not a question.”           /* 14 */
    End /* Select */                                    /* 15 */
  End num                                               /* 16 */
  Say “”                                                /* 17 */
  Say “Goodbye.”                                        /* 18 */
  Return 0                                              /* 19 */
/* Get answers from file */                             /* 20 */
Get_Answers:                                            /* 21 */
  qerc = Stream(file, ‘Command’, ‘Query Exists’)        /* 22 */
  orrc = Stream(file, ‘Command’, ‘Open Read’)           /* 23 */
  If qerc = ‘’ | orrc \= ‘READY:’ Then Do               /* 24 */
    Say “Unable to access” file“. Terminating.”         /* 25 */
    Exit 2                                              /* 26 */
  End                                                   /* 27 */
  Do i = 1 While Lines(file) > 0                        /* 28 */
    ans.i = Linein(file)                                /* 29 */
  End i                                                 /* 30 */
  ans.0 = i - 1                                         /* 31 */
  Call Stream file, ‘Command’, ‘Close’                  /* 32 */
  Return 0                                              /* 33 */
/* Provide Help */                                      /* 34 */
Help:                                                   /* 35 */
  Say “Enter as many yes/no questions as you wish.”     /* 36 */
  Say “Remember: a valid question must end with a”      /* 37 */
  Say “question mark.  All answers guaranteed to be”    /* 38 */
  Say “accurate, or your money back.”                   /* 39 */
  Say “To repeat this help message, enter ‘help’.”      /* 40 */
  Say “To terminate this program, enter ‘quit’.”        /* 41 */
  Return 0                                              /* 42 */
```

variable is the fundamental limit on the length of a variable name, 250 characters. The best I can figure, that allows you to create a compound variable that will handle 91 dimensions! That ought to take care of the most elaborate spreadsheet you can come up with. Remember, each of those 91 dimensions could contain millions of separate values, memory permitting. I have a hard time visualizing anything beyond the fourth dimension, myself.

Back to our problem. Listing 11 (lines 28-30) sets up a Do-loop where the variable "i" is methodically stepped from "1" to however many lines there are in the answers file. You could have three or three thousand answers, and this loop will build an array only as big as necessary. This is a very simple array: the elements are 1-based monotonically-increasing integers. That's a fancy way of saying that the first answer is found in "ans.1" and the second answer in "ans.2", et seqq. By convention (there's certainly nothing in Rexx that requires it), a simple list such as this usually will have its upper bound stored in the zero-th element. We do this on line 31. We have to subtract one from the index variable "i" because the Do-loop increments it before testing the "While" expression, which makes "i" one greater than the actual number of answers in the array.

So much for building an array. How do you use it? Look at line 13. We have totally eliminated the entire subprocedure that was necessary to select the random answer, and replaced it with a compound variable name.

This is a very simple array. Nothing says that the elements of the compound variable must start at "1", or be monotonically-increasing, or integers, or even numbers for that matter. Compound variables allow you to create an associative array, where the various dimensions are different characteristics associated with the data stored in it. For example, you could create a marketing contact database using only first names, hair color, type of car, and favorite beer. You might have a bunch of "Bob"s you deal with, but there probably isn't more than one that has "thinning-red" hair, drives a "Saturn", and drinks "Coors". If there is, simply add another category to distinguish between them. That's a silly example, but it points out how well associative arrays handle "messy data" where there isn't a readily-available number to use for an index.

Before we leave Listing 11, take a look at the 'If' instruction on line 24. We are testing the returned values from the two 'Stream' routines against their respective "failed" values. If 'Query Exists' can't find the file, it returns a null string. If 'Open' can't open the file, it returns an error message rather than the string "READY:". In both cases we want to issue a message and exit, but there's no sense in duplicating the code if we can do it in one 'If' instruction.

Rexx has LOGICAL OPERATORS that allow you to perform boolean operations on logical values. You've already seen one of them: the 'Not' prefix operator that inverts a logical value. The remaining logical operators are AND (&), OR (|), and EXCLUSIVE OR (&&). You might want to brush up on your truth tables, but they can be summarized as follows: "A & B" is true ("1") if and only if both of them are true, "A | B" is false ("0") if and only if both of them are false, and "A && B" is true if and only if they are different.

There are two very common mistakes using logical operators. One is forgetting that all Rexx operators are denoted by special characters, and coding something like "If x > 3 and y = 0 Then ...". The other is forgetting that a logical operator expects to see boolean values ("0" and "1"), and coding something like "If x = 4 | 8 Then ...".

Screen 11 shows what this program can do.

<Screen 11>

```
[D:\]rexx11
Enter as many yes/no questions as you wish.
Remember: a valid question must end with a
question mark.  All answers guaranteed to be
accurate, or your money back.
To repeat this help message, enter ‘help’.
To terminate this program, enter ‘quit’.

What is question number 1?
Is Rexx the language of the future?
Outlook good.

What is question number 2?
Should I buy a PowerPC?
Signs point to yes.

What is question number 3?
Do you believe in magic?
Better not tell you now.

What is question number 4?
quit

Goodbye.
```

ALL THE BELLS AND WHISTLES

As promised, here is the Magic 8-Ball program. This is where we introduce PARSE, arguably one of the most powerful Rexx keyword instructions. To parse a string means to break it up into substrings. We've been using it all along. Every time we 'Pull'ed a line from the keyboard, we parsed it into the variables in our template. 'Pull' is simply shorthand for PARSE UPPER PULL. Remember when we discussed the fact that 'Pull' always uppercased the string before assigning it to variables? The UPPER keyword causes that. You say you're working with mixed-case data, and don't want 'Pull' to do that? Don't use 'Pull'; use the more explicit 'Parse Pull'. If you don't say 'Upper', Rexx will leave your string as is. We want our string in all caps for ease of comparison, so we

<Listing 12>

```
/*============ Executive Decision Maker ===============*//*  1 */
/* Operation: Enter as many yes/no questions as you *//*  2 */
/*   wish.  Remember: a valid question must end with*//*  3 */
/*   a question mark.  All answers are guaranteed to*//*  4 */
/*   be accurate, or your money back.               *//*  5 */
/* Commands:                                        *//*  6 */
/*   ‘help’ - provides helpful information          *//*  7 */
/*   ‘quit’ - terminates program                    *//*  8 */
/* Requires:                                        *//*  9 */
/*   answer file - thisfilename’.ans’               *//* 10 */
/*------------------------*//* 11 */
  Parse Source . . me .                                 /* 12 */
  Parse Value Reverse(me) With . ‘.’ em                 /* 13 */
  ansdata = Reverse(em)’.ans’                           /* 14 */
  Call Get_Answers ansdata                              /* 15 */
  Call Help me                                          /* 16 */
  Do num = 1                                            /* 17 */
    Say “”                                              /* 18 */
    Say “What is question number” num”?”                /* 19 */
    Parse Upper Pull quest                              /* 20 */
    answer = Random(1, ans.0)                           /* 21 */
    Select                                              /* 22 */
      When quest = ‘QUIT’       Then Leave num          /* 22 */
      When quest = ‘HELP’       Then Call Help me       /* 24 */
      When Right(quest,1) = ‘?’ Then Say ans.answer     /* 25 */
      Otherwise Say “That is not a question.”           /* 26 */
    End /* Select */                                    /* 27 */
  End num                                               /* 28 */
  Say “”                                                /* 29 */
  Say “Goodbye.”                                        /* 30 */
  Return 0                                              /* 31 */
/* Get answers from file */                             /* 32 */
Get_Answers:                                            /* 33 */
  Procedure Expose ans.                                 /* 34 */
  Parse Arg file .                                      /* 35 */
  qerc = Stream(file, ‘Command’, ‘Query Exists’)        /* 36 */
  orrc = Stream(file, ‘Command’, ‘Open Read’)           /* 37 */
  If qerc = ‘’ | orrc \= ‘READY:’ Then Do               /* 38 */
    Say “Unable to access” file”. Terminating.”         /* 39 */
    Exit 2                                              /* 40 */
  End                                                   /* 41 */
  Do i = 1 While Lines(file) > 0                        /* 42 */
    ans.i = Linein(file)                                /* 43 */
  End i                                                 /* 44 */
  ans.0 = i - 1                                         /* 45 */
  Call Stream file, ‘Command’, ‘Close’                  /* 46 */
  Return 0                                              /* 47 */
/* Provide Help */                                      /* 48 */
Help:                                                   /* 49 */
  Procedure                                             /* 50 */
  Parse Arg file .                                      /* 51 */
  num = Stream(file, ‘Command’, ‘Open Read’)            /* 52 */
  If num = ‘READY:’ Then Do                             /* 53 */
    Do Until Left(helpline, 3) = ‘—’                   /* 54 */
      fileline = Linein(file)                           /* 55 */
      Parse Var fileline ‘/*’ helpline ‘*/’ .           /* 56 */
      Say helpline                                      /* 57 */
    End                                                 /* 58 */
    Call Stream file, ‘Command’, ‘Close’                /* 59 */
  End                                                   /* 60 */
  Else Say num “opening” file                           /* 61 */
  Return 0                                              /* 62 */
```

use the full form on line 20 of Listing 12.

As you might have gathered, there are a number of places from which you can obtain a string to be 'Parse'd. On line 12, we PARSE SOURCE to extract information about our program. The information varies by operating system, but on OS/2 it returns three words. The first word is the name of the operating system ("OS/2"). The second word indicates whether this program was invoked as a "COMMAND", "SUBROUTINE", or "FUNCTION". The third word is the fully-qualified file specification of this program ("D:\rexx12.cmd"). Other platforms provide more information, but the first three words are consistent. The third word is what we need so we can let the program find its own answer file. The first thing we must do is get rid of the ".cmd" extension on the program's filespec. Unfortunately, there is no single function that will "remove all the characters following the last period", so we'll just parse it off.

The 'Parse' template is a very powerful search-and-extract tool. In addition to parsing by words (blank delimited strings), you can parse the data by character position (absolute and relative), delimited by specified strings, or even let the data itself tell the template where the data is. Here, we want to take everything from the beginning of the filespec to the last period. Figuring out which is the last period in an arbitrary filespec is tough. Finding the first one is easy, though.

Quick solution: reverse the string end-for-end and look for the "first" period. This sounds like a two-step process, but we can tell Rexx to parse the result of an expression, as in line 13. Rexx will evaluate the expression between the keywords 'Value' and 'With', and then parse the resulting string according to the template. A literal string in the template indicates a delimiter. Our template says to throw away everything up to the first delimiter we specify (in this case, the literal string '.'), then take everything after the delimiter and put it in the variable "em". All we have to do now is to re-reverse the remaining part of the filespec, and append the proper extention (".ans").

In this case, we wanted the entire filespec, drive letter and all. If all you want is the filename without the path, add '\' to the end of the template. The variable 'me' will then contain only those characters between the last backslash and the last period.

This filespec is then passed as an argument to the "Get_Answers" subprocedure. How do you suppose "Get_Answers" retrieves that argument? With the PARSE ARG instruction, of course (line 35). All flavors of the 'Parse' instruction manipulate a string according to the template you specify. The only difference between them is where they get the string to be parsed. 'Parse Arg' gets its string from the arguments passed to the currently executing procedure. This could be a subprocedure like "Get_Answers", or a subprocedure it invoked, or even the top-level Rexx program itself. After all, it's just a subroutine of OS/2. When you invoke a Rexx program from the command line, any string following the program name is passed as an argument to the program. The same thing goes for anything you put in the "Parameters" field on the "Program" page of the "Settings" notebook for an object. By the way, ARG is shorthand for 'Parse Upper Arg'. Because 'Arg' and 'Pull' are the only 'Parse' shorthands, and they don't explicitly indicate that the data is being uppercased, I don't use them (except for pedagogical purposes).

Why bother putting the argument on the 'Call' and using 'Parse Arg' to get it? Can't "Get_Answers" just use the variable "ansdata" as it did before? Not since we started the subprocedure with PROCEDURE. The 'Procedure' instruction isolates the variables in the subprocedure from all the others. Look at the "Help" subroutine (lines 49-62). We weren't paying real good attention, and assigned the return code from the 'Stream' routine to the variable "num". No big deal, except that the main program happens to be using that same variable name as the index variable of the Do-loop. This will cause the program to exhibit somewhat baffling behavior: it will work fine until you ask for help, which it will provide, but then it will blow up when it tries to prompt you for the next question. With 'Procedure', you don't have to keep track of every variable name you've used, just the ones in a particular subprocedure. Your program will be more robust, less error-prone, and errors that do occur will be limited to a smaller region of your code.

The data flowing into and out of a subprocedure is now well defined: data comes in via 'Parse Arg' and goes back via 'Return'. That's the upside, and it's a Good Thing. The drawback is that the data flowing into and out of a subprocedure MUST be well defined, and that can be a pain. Specifying 'Procedure' as the first instruction of a subprocedure makes all of its variables "local" and hides all other variables from it. By the same token, no other routine can see or modify the subprocedure's variables. It certainly fixed our problem in the 'Help' routine. Now Rexx has two "num" variables, but only one can be in use at a time.

When you're using a lot of variables you'll want to use 'Procedure'. Of course, if you have a subprocedure that must access a lot of variables, it can be a real chore to pass them all as arguments everytime you invoke the routine. The 'Expose' keyword on the 'Procedure' instruction opens a small window between the two variable pools. Any variable specified in the blank-delimited list following 'Expose' will remain global, and be accessible by both routines. Everything else is still hidden.

Speaking of the 'Help' subroutine, what's going on there? First, notice that we now have a program header we can be proud of. Production programs often have more documentation in such prologs than they have executable statements. Unfortunately, you often end up writing much the same information into the "Help" routine for the user. Wouldn't it be nice if you could use the same documentation for both? Since it's embedded within the program, it stands a better chance of being current, too. Now that we know how to read lines from a file, and how to find out what our filename is, what's to keep us from reading our own program and extracting the documentation directly from the prolog? All we need is some way to know where the header ends, and a way to extract the text from the comments.

The latter is easy: use a template that says "look for anything between a '/*' and an '*/' and put it into a variable. But where do we get the string to be parsed? Where did you put it when you read it in with 'LineIn'? A variable, you say? Then let's do a PARSE VAR and specify that variable name.

Knowing when to quit is not one of my strong suits, but it's easy in this case. After writing each line of help text to the screen, we check the left-most three characters to see if they match those of the last line of the prolog. If so, we're done.

Screen 12 shows the results.

<Screen 12>

```
=========== Executive Decision Maker =============
 Operation: Enter as many yes/no questions as you
   wish.  Remember: a valid question must end with
   a question mark.  All answers are guaranteed to
   be accurate, or your money back.
 Commands:
   'help' - provides helpful information
   'quit' - terminates program
 Requires:
   answer file - thisfilename'.ans'
__________________________________________________

What is question number 1?
Will the Voyager crew ever make it home?
Cannot predict now.

What is question number 2?
Will I become rich and famous teaching Rexx?
My reply is no.

What is question number 3?
Either one?
Very doubtful.

What is question number 4?
quit

Goodbye.
```

ISSUING COMMANDS

Believe it or not, the impetus to design Rexx came from the lack of a decent batch language on VM, back in the late 70's. On the other hand, VM was in deep clover compared to the batch language that would be supplied by that little software company (which shall remain nameless) that gave us DOS. At least the VM batch language had a way to get input from the keyboard! Rexx was designed to be used as a 'glue' language to interface to application programs and the operating system. Big deal, you say. "I can issue OS/2 commands from my .BAT file." Sure you can. But can you retrieve the output from that command and process it, or is it simply dumped to the screen? Processing operating system commands in Rexx is trivially easy. Better yet, Rexx performs expression evaluation before sending it off, so you can dynamically create the command.

```
<Listing 13>

/* rexx13.cmd: Determine total bytes used by folder *//*  1 */
  Trace Off                                           /*  2 */
  Address CMD                                         /*  3 */
  lf = ‘0A’x                                          /*  4 */
  diropts = ‘/S /A /W’                                /*  5 */
  Parse Source . . me .                               /*  6 */
  Parse Value Reverse(me) With . ‘.’ me ‘\’ .         /*  7 */
  me = Reverse(me)                                    /*  8 */
  Parse Arg path .                                    /*  9 */
  newdir = Directory(path)                            /* 10 */
  If newdir = ‘’ Then Call Error ‘Unable to find’ lf path
  ‘@DIR’ diropts ‘| RXQUEUE /LIFO’                    /* 12 */
  If Rc \= 0 Then Call Error ‘DIR’ diropts ‘| RXQUEUE’/* 13 */
  Parse Pull free .                                   /* 14 */
  fsize = free / 1024                                 /* 15 */
  If fsize < 1024 Then fsize = Format(fsize, 4, 2)’K’ /* 16 */
  Else fsize = Format(fsize / 1024, 4, 2)’M’          /* 17 */
  Parse Pull files . used .                           /* 18 */
  usize = used / 1024                                 /* 19 */
  If usize < 1024 Then usize = Format(usize, 4, 2)’K’ /* 20 */
  Else usize = Format(usize / 1024, 4, 2)’M’          /* 21 */
  sizemsg = newdir lf ,                               /* 22 */
            files ‘files’ lf ,                        /* 23 */
            usize ‘(‘used’) used’ lf,                 /* 24 */
            fsize ‘(‘free’) free’                     /* 25 */
  Call RxMessageBox sizemsg, me, ‘OK’, ‘INFORMATION’  /* 26 */
  Return 0                                            /* 27 */
Error:                                                /* 28 */
  Parse Arg emsg                                      /* 29 */
  Call RxMessageBox emsg, me, ‘CANCEL’, ‘ERROR’       /* 30 */
  Exit Rc                                             /* 31 */
```

How does Rexx distinguish between a command and a Rexx instruction? Easy. First, if it's a comment or a blank line it is discarded. If it is a label, its position is noted, and it is discarded. If the first thing on the line looks like a variable, and the second thing is an equal sign, Rexx takes it for an assigment instruction. If the second thing is not '=', Rexx checks the first thing to see if it's a Rexx keyword. If so, Rexx sends the instruction off to that keyword's processing routine. If not, then Rexx doesn't recognize it as a Rexx instruction, so it gets sent to the outside environment (usually the OS/2 command handler). There can be a multiplicity of outside environments: EPM, AmiPro, SQL, ZOC, ISPF, all sorts of programs have direct Rexx interfaces. Even those that don't can be driven by Rexx if they have any sort of command line capability.

My last example, Listing 13, solves one of those nagging little problems that occur when the operating system doesn't do things quite the way you'd like. Specifically, you want to know how much space is taken up by a particular directory, including all of its subdirectories. Yes, you can do a "DIR /S" to get the information, but it's at the bottom of 42 screens of output. Not only that, but you've got to open up a command window and key in the full filespec, if you don't happen to be in that directory at the moment. Wouldn't it be nice if you could just drag a folder to an icon and see the answer in a pop-up window? Of course it would.

Since this is my last example, there are a few miscellaneous items I want to demonstrate. Let's start at the top. The TRACE instruction is one of those things that makes programming in Rexx such a joy. If your programs run the first time, you won't be very interested in this. I, on the other hand, use it a lot. The Rexx error messages are pretty good about explaining what the problem is, but sometimes it's just not enough. What you need is for Rexx to step through the execution of your program and tell you what it is doing along the way.

The 'Trace' keyword instruction does exactly that and more. There are nine kinds of traces in OS/2, but you will probably use only three. 'Trace Off' is the most common, and tells Rexx not to trace anything. 'Trace Results' tells Rexx to list each instruction as it is encountered, then write one more line (usually) that indicates the end result of executing that instruction. Certain Rexx instructions do so much that a single trace line is insufficient; these instructions may create several lines of trace output. Tracing begins when Rexx encounters the 'Trace' instruction, and ends where you turn tracing off (or reach the end of execution). You can sprinkle 'Trace's throughout your program to see what's going on in various places. Unfortunately, the default trace setting is not 'Off', so it's a good idea to set it explicitly at the start of your program. Unless you are tracing just a few lines only, the trace output will quickly scroll off the screen. The trace output is written to "STDOUT:", so you can redirect it to a file by appending "2> trace.out" to the command that invokes your program. Alternatively, you can run it under PMREXX, and save the results to a file. Screens 13a and 13b show you portions of a 'Results' trace and an 'Intermediates' trace of Listing 13.

For a real kick in the head, put a "?" before the trace type (eg. "Trace ?Results"). This will drop you into INTERACTIVE TRACE, where Rexx will stop after every instruction and wait for input. This gives you the opportunity to modify or display variables, insert instructions, and even to back up and re-execute the previously traced instruction!

The ADDRESS instruction on line 3 directs Rexx to send commands to the OS/2 command handler. If you wanted to send commands to, say TCP/IP, you would specify the name of that command interface. You can change 'Address' destinations at any point in your program, flip-flop back and forth between two destinations, and send a single command to a particular destination without disturbing the current 'Address' setting.

On line 4 we assign a hexadecimal value to a variable. Hex and binary constants are specified as a literal string containing the appropriate values, immediately followed by an 'x' or 'b', respectively. Rexx has a suite of built-in functions that allow you to convert among hex, binary, and character encodings. We need this value (linefeed) to cause our message text to start a new line in the pop-up window.

The DIRECTORY routine on line 11 changes the current directory to the one passed as an argument, and returns its full filespec if successful. Unlike the 'CD' command, it will even change disks. If it cannot find the directory you specify, it returns a null string, which is what we check for in line 11.

Line 12 is where we actually issue the "DIR" command. The "@" tell OS/2 that you don't want to see the "DIR" command itself, echoed on the screen. If all we wanted was to save ourselves some keystrokes, the second literal on the line would not have been necessary. Rexx would have concatenated the contents of the "diropts" variable to the literal "DIR" and sent "DIR /S /A /W" off to OS/2 for processing. The output of the "DIR" command would have then scrolled off our screen as usual. When you tell OS/2 to pipe the output of the "DIR" command into the RXQUEUE command, the output is put in to a queue for Rexx to access. In this case, all we want is the last two lines of the output, so we direct 'RXQUEUE' to place the lines on the queue in last-in-first-out order.

All commands issue a return code, usually indicating some sort of completion status. Rexx puts return codes into the special variable RC, where we can check to see that all went well. The third special variable, in case you're wondering, is SIGL. 'Sigl' contains the line number from which control was passed when you invoke a subprocedure.

About now, you're probably thinking that I lied to you when I said that 'Parse Pull' (or 'Pull') pulled a line from the keyboard. It was just a little one. 'Parse Pull' really does read from the keyboard, but only if the queue is empty. Since that's been the case so far, you didn't notice. The 'Parse Pull' on line 14 retrieves the last line of the "DIR" output (the number of bytes used). This value is divided by 1024 to get the number of 'K', and if the quotient is as much as 1024, divides it again to get the value in megabytes. Whatever value we end up with is then FORMATted with four digits before the decimal point, and two after it. The same process is used on the next-to-last line in the queue, which also contains the number of files encountered. We then create our output message line which will span four lines. This assignment statement is too long to fit on one line of my program, so I continue it onto the next three lines. This is done by placing a comma (,) as the last character (comments don't count) of each line except for the last. This only affects the formatting of the Rexx program for readability; the "lf" value imbedded in the message text is what controls how many lines are displayed in the window.

<Screen 13a>

```
     3 *-*  Address CMD;
     4 *-*  lf = ‘0A’x;
       >>>    “?”
     5 *-*  diropts = ‘/S /A /W’;
       >>>    “/S /A /W”
     6 *-*  Parse Source . . me .;
       >.>    “OS/2”
       >.>    “COMMAND”
       >>>    “F:\chip\rexxclass\rexx13.cmd”
       >.>    “”
     7 *-*  Parse Value Reverse(me) With . ‘.’ me ‘\’ .;
       >>>    “dmc.31xxer\ssalcxxer\pihc\:F”
       >.>    “dmc”
       >>>    “31xxer”
       >.>    “ssalcxxer\pihc\:F”
     8 *-*  me = Reverse(me);
       >>>    “rexx13”
     9 *-*  Parse Arg path .;
       >>>    “”
       >.>    “”
    10 *-*  newdir = Directory(path);
       >>>    “F:\chip\rexxclass”
    11 *-*  If newdir = ‘’;
       >>>    “0”
    12 *-*  ‘@DIR’ diropts ‘| RXQUEUE /LIFO’;
       >>>    “@DIR /S /A /W | RXQUEUE /LIFO”
    13 *-*  If rc <> 0;
       >>>    “0”
    14 *-*  Parse Pull free .;
       >>>    “2598912”
       >.>    “bytes free”
    15 *-*  fsize = free / 1024;
       >>>    “2538”
    16 *-*  If fsize < 1024;
       >>>    “0”
    16 *-*  Else;
    17 *-*    fsize = Format(fsize / 1024, 4, 2)‘M’;
       >>>    “   2.48M”
    18 *-*  Parse Pull files . used .;
       >>>    “127”
       >.>    “file(s)”
       >>>    “775708”
       >.>    “bytes used”
    19 *-*  usize = used / 1024;
       >>>    “757.527344”
    20 *-*  If usize < 1024;
       >>>    “1”
    20 *-*  Then;
    20 *-*    usize = Format(usize, 4, 2)‘K’;
       >>>      “ 757.53K”
    22 *-*  sizemsg = newdir lf files ‘files’ lf usize ‘(‘used’) used’ lf fsize ‘(‘free’) free’;
       >>>    “F:\chip\rexxclass ? 127 files ? 757.53K (775708) used ?    2.48M (2598912) free”
    26 *-*  Call RxMessageBox sizemsg, me, ‘OK’, ‘INFORMATION’;
       >>>    “1”
    27 *-*  Return 0;
       >>>    “0”
```

<Screen 13b>

```
     3 *-*   Address CMD;
...(snipped)...
       >>>     “759.34082”
    20 *-*   If usize < 1024;
       >V>     “759.34082”
       >L>     “1024”
       >O>     “1”
       >>>     “1”
    20 *-*   Then;
    20 *-*     usize = Format(usize, 4, 2)’K’;
       >V>       “759.34082”
       >L>       “4”
       >L>       “2”
       >F>       “ 759.34”
       >L>       “K”
       >O>       “ 759.34K”
       >>>       “ 759.34K”
    22 *-*   sizemsg = newdir lf files ‘files’ lf usize ‘(‘used’) used’ lf fsize ‘(‘free’) free’;
       >V>     “F:\chip\rexxclass”
       >V>     “?”
       >O>     “F:\chip\rexxclass ?”
       >V>     “128”
       >O>     “F:\chip\rexxclass ? 128”
       >L>     “files”
       >O>     “F:\chip\rexxclass ? 128 files”
       >V>     “?”
       >O>     “F:\chip\rexxclass ? 128 files ?”
       >V>     “ 759.34K”
       >O>     “F:\chip\rexxclass ? 128 files ?  759.34K”
       >L>     “(“
       >O>     “F:\chip\rexxclass ? 128 files ?  759.34K (“
       >V>     “777565”
       >O>     “F:\chip\rexxclass ? 128 files ?  759.34K (777565”
       >L>     “) used”
       >O>     “F:\chip\rexxclass ? 128 files ?  759.34K (777565) used”
       >V>     “?”
       >O>     “F:\chip\rexxclass ? 128 files ?  759.34K (777565) used ?”
       >V>     “   2.48M”
       >O>     “F:\chip\rexxclass ? 128 files ?  759.34K (777565) used ?    2.48M”
       >L>     “(“
       >O>     “F:\chip\rexxclass ? 128 files ?  759.34K (777565) used ?    2.48M (“
       >V>     “2596864”
       >O>     “F:\chip\rexxclass ? 128 files ?  759.34K (777565) used ?    2.48M (2596864”
       >L>     “) free”
       >O>     “F:\chip\rexxclass ? 128 files ?  759.34K (777565) used ?    2.48M (2596864) free”
       >>>     “F:\chip\rexxclass ? 128 files ?  759.34K (777565) used ?    2.48M (2596864) free”
    26 *-*   Call RxMessageBox sizemsg, me, ‘OK’, ‘INFORMATION’;
       >V>     “F:\chip\rexxclass ? 128 files ?  759.34K (777565) used ?    2.48M (2596864) free”
       >V>     “rexx13”
       >L>     “OK”
       >L>     “INFORMATION”
       >>>     “1”
    27 *-*   Return 0;
       >L>     “0”
       >>>     “0”
```

The RxMessageBox routine causes a pop-up window to display with our message in it. We can choose a title for the box (the name of our program), select a variety of pushbuttons to enable, and chose an icon to be displayed.

Don't try to run this from the command line. 'RxMessageBox' needs to have access to the Presentation Manager to work. From the command line, invoke PMREXX and pass the name of this program (and its argument) as an argument to PMREXX. Alternatively, create a Program Reference object and open the Program tab of its Settings notebook. Specify "drive:\OS2\PMREXX.EXE" (where "drive" is your OS/2 system drive) as the "Path and file name". In the "Parameters" box, enter the filespec of rexx13.cmd (or whatever you decide to call it). If you drop a folder onto this icon, you should see a pop-up window that tells you how much storage that folder requires.

CONCLUSION

Well, that should keep you busy exploring Rexx for a while. When you have questions, there are a number of excellent Rexx texts on the market to choose from, including the definitive "The Rexx Language: A Practical Approach To Programming", by M.F. Cowlishaw, the creator of the language. Many of the authors are here this week, in fact.

The Rexx Language Association is an international association of Rexx programmers, implementors, and vendors, that I encourage you to join. The RexxLA sponsors the annual Rexx Symposium, where Rexx enthusiasts from all over the world gather to explore anything and everything Rexx-ish. If you have Internet access, the place to be is 'comp.lang.rexx' where you'll find everyone from beginners to mavens to Mike Cowlishaw. There are several World-Wide Web pages devoted to Rexx, but you might want to start with Mike's, which is at "http://rexx.hursley.ibm.com/rexx/".

If I can be of any further help, you can send me e-mail at chip@acm.org and I'll do my best to give you a hand.

PRO 0306

Introduction to SOM for C Programmers

Roger Sessions

Copyright Notice

- This entire presentation is copyright by ObjectWatch, Inc., Austin, Texas. It may not be redistributed, in whole or in part, without permission.

Overview of Talk

- Introduction to Objects
  - Objects in C
  - Objects in SOM
- OO Features of SOM
  - Inheritance
  - Polymorphism
- Object Distribution
- Other nice SOM features

Definition of Class

- A relationship between a data template and a set of behaviors
- Example of a class
  - The animal class
- What is an animal?
  - an animal has a name
  - you can tell the animal its name
  - you can ask the animal its name

C Definition of animal (animal.h)

```
struct animalType {
  char name[100];
};
typedef struct animalType animal;

void setName(animal *self, char *newName);
char *getName(animal *self);

animal *animalNew();
```

C animal Implementation (animal.c)

```
#include <string.h>
#include <stdio.h>
#include "animal.h"
void setName(animal *self, char *newName)
{
  strcpy(self->name, newName);
}
char *getName(animal *self)
{
  return self->name;
}
animal *animalNew()
{
  return (animal *) malloc(sizeof(animal));
}
```

animal Client code

```
#include "animal.h"
void main()
{
  animal *pooh;
  pooh = animalNew();
  setName(pooh, "Pooh Bear");
  /* ... */
  printf("My name is: %s\n", getName(pooh));
}
```

Terminology

- animal is a *class*
- pooh is an *object* of class animal
- The methods of the animal class are
  - setName
  - getName
- We *invoke* methods on a *target object*
  - setName(pooh, "Pooh Bear");
- We *instantiate* objects of a given class
  - Analagous to allocating memory
  - Classes have *instantiator* methods - newAnimal())

Encapsulation

- animal clients need to know...
  - the name of the class (animal)
  - the names and prototypes of the methods (getName())
  - the semantics of the methods
- animal clients do not need to know
  - the data in an animal
  - how the methods are implemented
- The data structure and implementation of the methods are *encapsulated* in the class
- Encapsulation protects clients from changes in class implementation

Original animal class

```
struct animalType {
  char name[100];
  char food[100];
};
typedef struct animalType animal;

void setName (animal *self, char *newName)
{
 strcpy(self->name, newName);
}
```

New animal implementation

```
struct animalType {
 char *name;
};
typedef struct animalType animal;

void setName(animal *self, char *newName)'
{
 if(self->name) free(self->name);
 self->name = (char *) malloc(strlen(newName + 1);
 strcpy(self->name, newName);
}
```

Client Code is Unchanged!

```
#include "animal.h"
void main()
{
  animal *pooh;
  pooh = animalNew();
  setName(pooh, "Pooh Bear");
  /* ... */
  printf("My name is: %s\n", getName(pooh));
}
```

Comparing C and SOM Classes

- Both have data associated with objects
- Both have methods associated with objects
- Both implement the methods with standard C code

Compare the C definition...

```
struct animalType {
  char name[100];
};
typedef struct animalType animal;

void setName(animal *self, char *newName);
char *getName(animal *self);

animal *animalNew();
```

...to this (simplified) SOM definition:

```
interface animal {

  void setName(char *newName);
  char *getName();

  implementation {
    char name[100];
  };

};
```

Compare the C implementation...

```
#include <string.h>
#include <stdio.h>
#include "animal.h"
void setName(animal *self, char *newName)
{
  strcpy(self->name, newName);
}
char *getName(animal *self)
{
  return self->name;
}
animal *animalNew()
{
  return (animal *) malloc(sizeof(animal));
}
```

... to this (simplified) SOM Implementaton

```
#include <string.h>
#include <stdio.h>
#include "animal.ih"
void setName(animal *somSelf, char *newName)
{
  strcpy(somSelf->name, newName);
}
char *getName(animal *somSelf)
{
  return somSelf->name;
}
```

Compare the C client code...

```
#include "animal.h"
void main()
{
  animal *pooh;
  pooh = animalNew();
  setName(pooh, "Pooh Bear");
  /* ... */
  printf("My name is: %s\n", getName(pooh));
}
```

... to this SOM Client Code

```
#include "animal.h"
void main()
{
 animal pooh;
 pooh = animalNew();
 _setName(pooh, "Pooh Bear");
 /* ... */
 printf("My name is: %s\n", _getName(pooh));
}
```

Both give the same ouput

```
My name is: Pooh Bear
```

Why is SOM better than C?

- Full Support for Encapsulation
- Inheritance
- Polymorphism
- An Advanced Object Model
- Compatability with CORBA (from Object Management Group)
- And The Most Important Reason of All...

Inheritance

- animals can have a name
- Consider dogs
  - dogs can have a name
  - dogs can make some noise
- Notice: dogs are animals that make noise
- We can derive a new class, dog, from an existing class, animal

Definition of dog

```
#include <animal.idl>
interface dog : animal {
  string getBark();
  string barkString();
};
```

Implementation of dog

```
string  getBark(dog somSelf,  Environment *ev)
{
   string myBarkString;
   string myBark;

   myBarkString = _barkString(somSelf, ev);
   myBark = (string) SOMMalloc(strlen(myBarkString) + 1);
   strcpy(myBark, myBarkString);
   return myBark;
}
string  barkString(dog somSelf,  Environment *ev)
{
   static string myBarkString = "Unknown Dog Noise\n";
   return myBarkString;
}
```

dog Client

```
#include "dog.h"
void main()
{
  snoopie dog;
  snoopie = dogNew();
  _setName(snoopie "Snoopie");
  /* ... */
  printf("My name is: %s\n", _getName(snoopie));
  printf("I say: %s\n", _getBark(snoopie));
}
```

Client Output

```
My name is: Snoopie
I say: Unknown Dog Noise
```

Inheritance

- Allows derived classes to reuse code of base class
- Improvements to base classes are automatically picked up be derived classes

Polymorphism

- littleDog and bigDog classes
  - Just like dog, with a different bark
- Inheritance, with a twist
  - We want to modify functonality, not add functonality

bigDog

```
#include <dog.idl>
interface bigDog : dog {
  implementation {
    override: barkString;
  };
};
```

```
string barkString(bigDog somSelf,  Environment *ev)
{
    static string myBarkString = "Woof Woof\nWoof Woof\n\n";
    return myBarkString;
}
```

littleDog

```
#include <dog.idl>
interface littleDog : dog {
  implementation {
    override: barkString;
  };
};
```

```
string  barkString(littleDog somSelf,  Environment *ev)
{
    static string myBarkString = "woof woof\n";
  littleDogMethodDebug("littleDog","getBark");
    return myBarkString;
}
```

Client Code

```
#include "dog.h"
void main()
{
  snoopie dog;
  littleDog toto;
  bigDog lassie;

  snoopie = dogNew();
  toto = littleDogNew();
  lassie = bigDogNew();
```

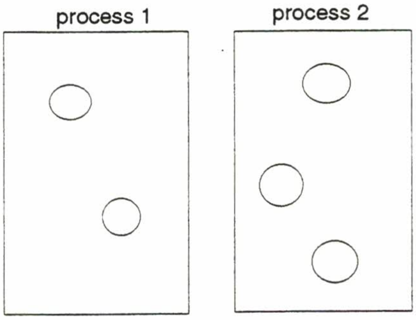
Client Code (continued)

```
 _setName(snoopie, "Snoopie");
_setName(toto, "Toto");
_setName(lassie, "Lassie");
  /* ... */
  printf("My name is: \n%s\n", _getName(snoopie));
  printf("I say: \n%s\n", _getBark(snoopie));

  printf("\nMy name is: %s\n", _getName(toto));
  printf("I say: %s\n", _getBark(toto));

  printf("\nMy name is: %s\n", _getName(lassie));
  printf("I say: \n%s\n", _getBark(lassie));
}
```

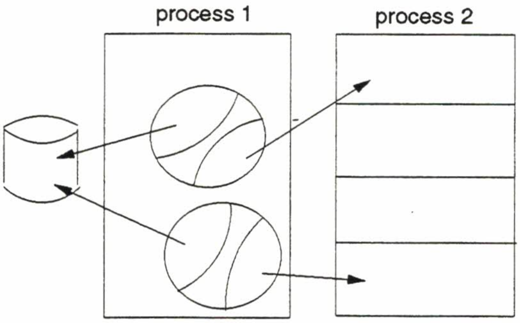
Client Output

```
My name is: Snoopie
I say:
Unknown Dog Noise

My name is: Toto
I say:
woof woof

My name is: Lassie
I say:
Woof Woof
Woof Woof
```

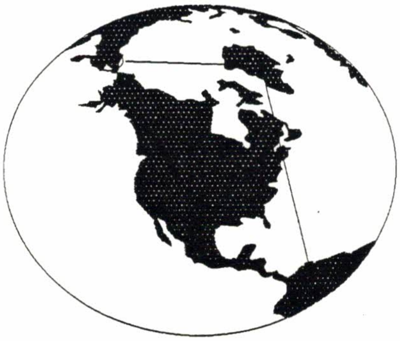
Objects in the 80s


process 1
process 2

Objects in the 90s


process 1
process 2

Objects in the 2000s



The Most Important SOM Feature:

- The Leading Distributed Object Technology
- IBM's implementation of the OMG CORBA specification
- Committed to an open object distribution standard

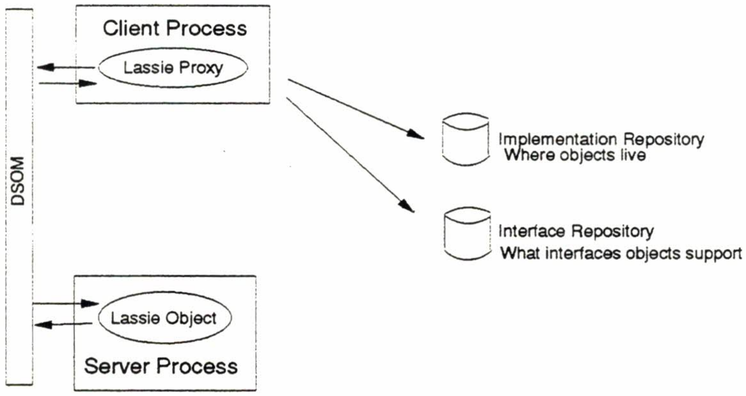
Scalability of Distribution

- Start with local objects
- Distribute objects in multiple processes
- Distribute objects across a network
- Distribute objects across a world
- ...

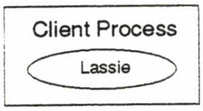
Networks Supported

- Single process (All)
- Shared memory (All)
- TCP/IP (AIX, OS/2, Windows)
- Netware IPX/SPX (AIX, OS/2, Windows)
- NetBIOS (OS/2)

SOM Architecture


DSOM
Client Process
Lassie Proxy
Implementation Repository
Where objects live
Interface Repository
What interfaces objects support
Lassie Object
Server Process

User View of Architecture



The Programming Process

- Define an interface in IDL
- Use IDL Compiler to generate language bindings
- Implement interface in language of choice
- Build implementation as a DLL
- Write client program to access object using language bindings
  - Client programming language independent of implementation language

Client Code (1/2)

```
#include <somd.h>
#include <dog.h>
#include <ldog.h>
#include <bdog.h>
#include <stdio.h>

void main()
{
 dog snoopie;
 littleDog toto;
 bigDog lassie;

  Environment *ev = SOM_CreateLocalEnvironment();
 SOMD_Init(ev);
```

Client Code (2/2)

```
snoopie = _somdNewObject(SOMD_ObjectMgr, ev, "dog", "");
 toto = _somdNewObject(SOMD_ObjectMgr, ev, "littleDog", "");
 lassie = _somdNewObject(SOMD_ObjectMgr, ev, "bigDog", "");

 _setName(snoopie, "Snoopie");
 _setName(toto, "Toto");
 _setName(lassie, "Lassie");
```

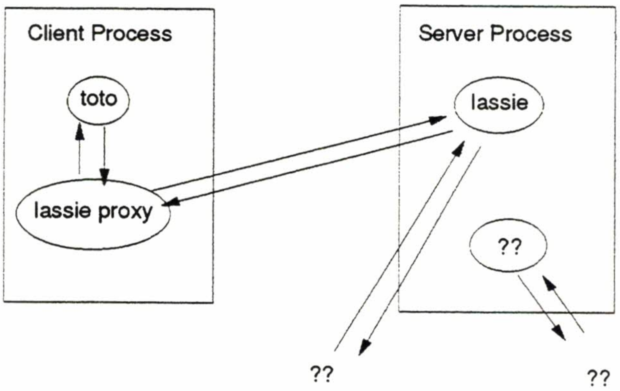
Client Code (continued)

```
/* ... */
printf("My name is: \n%s\n", _getName(snoopie));
printf("I say: \n%s\n", _getBark(snoopie));

printf("\nMy name is: %s\n", _getName(toto));
printf("I say: %s\n", _getBark(toto));

printf("\nMy name is: %s\n", _getName(lassie));
printf("I say: \n%s\n", _getBark(lassie));
```

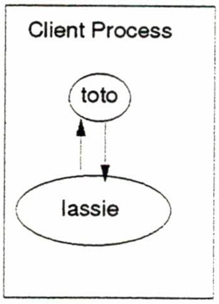
Client Code (continued)

```
    _somdDestroyObject(SOMD_ObjectMgr, ev, lassie);
    _somdDestroyObject(SOMD_ObjectMgr, ev, toto);
    _somdDestroyObject(SOMD_ObjectMgr, ev, snoopie);

    SOMD_Uninit(ev);
    SOM_DestroyLocalEnvironment(ev);
}
```

Distributed Object Architecture


Client Process
toto
lassie proxy
Server Process
lassie
??
??
??

Client View


Client Process
toto
lassie

Other Nice SOM Features

- Binary independence
- Advanced Object Model
  - Classes represented as objects
  - Metaclass framework
- Compatible with existing languages
  - C++ Direct To Som Compilers

Summary

- SOM is a system for implementing Distributed Objects
- Distributed Objects are the Future
- SOM Objects can be in another process or another country
- Fully compliant with the OMG CORBA specification for distributed object architectures

About the Author

Roger Sessions has spoken at dozens of conferences throughout the world and has written extensively about SOM and Object-Oriented Programming. He has given SOM tutorials at OOPSLA and other conferences and has consulted on many large scale SOM projects.

Roger was part of the IBM SOM development team from 1990-1995. He was the technical lead for the SOM Persistence Framework and a principal architect of the OMG's Persistent Object Service, an industry standard for Object Persistence based on the SOM Persistence Framework

In 1995, Roger left IBM to form ObjectWatch, Inc., a company specializing in consulting, training, and mentoring in the use of SOM, DSOM, and other aspects of Distributed Object Technology.

Roger is the author of three books. His most recent book, scheduled for release in late 1995, is ***Object Persistence: Beyond Object-Oriented Databases*** , an upfront and personal view of the OMG Persistence Specification. He has also written *Class Construction in C and C++; **Object-Oriented Fundamentals*** and ***Reusable Data Structures for C***. Roger has co-authored many articles in OS/2 publications.

He can be reached at roger@fc.net.

PRO 0307

Storing Objects in DB/2: The Ultimate Object Datastore

Guylaine Cantin
IBM

Caution

- This presentation covers ongoing work.
- This design is being presented to get customer feedback.
- The interfaces shown may change without warning.
- IBM has not yet committed to shipping any products based on the work shown here.

Agenda

- Background
- Why DB2?
- Overview of the OMG Persistence Model
- Plugging in your Datastore
- A case study of DB2
- Sample Client Code
- A family of OMG Services
- Summary
- Demo

Our Vision

- DB2 provides advanced database technology to objects
- SOM provides advanced object capabilities to DB2

DB2 + SOM = Ultimate Object Datastore !

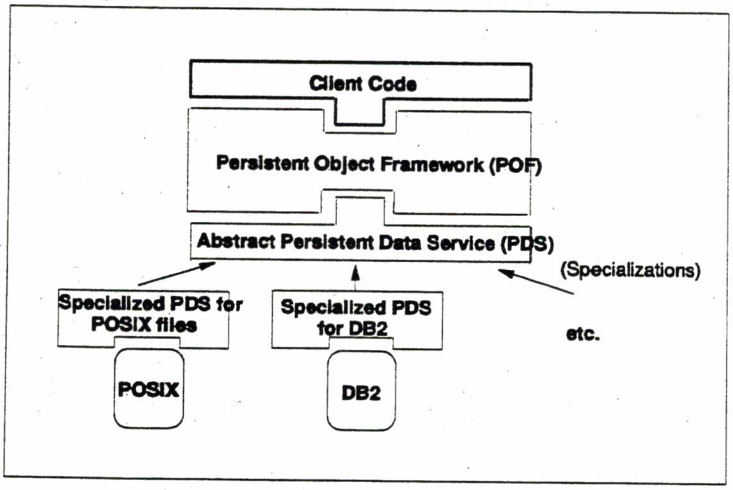
Value to Customer

- DB2 is Industry Standard object datastore
- DB2 customers get all the benefits of SOM
  - Language Neutrality
  - High Performance, transparent distributed object support
- Full compatibility with all existing DB2 corporate applications
- Ability to store objects in existing DB2 databases
- Plug compatible with tomorrow's OODBMS

Target Customer

- Needs the benefits of object technology
  - Faster application development
  - Reduced maintenance cost
  - Improved code quality
  - Increase code reuse
- Multi-Million dollar investment in existing applications
  - Must be able to access new object data
- Terabytes of critical data in RDBMS products
  - Must be accessible by new OO applications
- Data Spread across many datastores

OMG Persistence Model



Client Code

Persistent Object Framework (POF)

Abstract Persistent Data Service (PDS)

(Specializations)

Specialized PDS for POSIX files

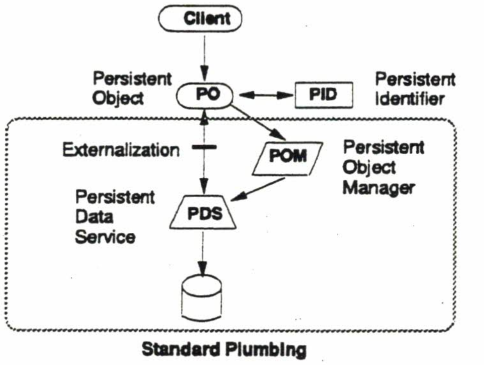
Specialized PDS for DB2

etc.

POSIX

DB2

High Level Architecture


Client
Persistent Object
PO
PID
Persistent Identifier
Externalization
POM
Persistent Object Manager
Persistent Data Service
PDS
Standard Plumbing

OMG Persistence Interfaces

- Persistent Identifier (PID)
  - specifies location of object's persistent data
- Persistent Object (PO)
  - optional client visible persistence interface
- Persistent Object Manager (POM)
  - ensures datastore independence
  - routes to a PDS specialized for a datastore
- Persistent Data Service (PDS)
  - encapsulates datastore specific code
- Streamable Protocol
  - mechanism for getting data in/out of the object

Plugging in Your Datastore

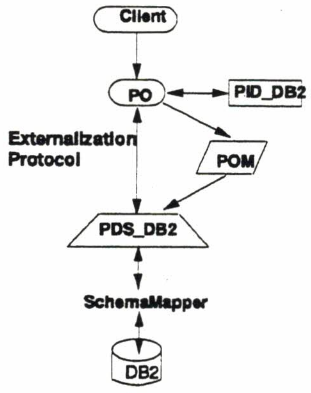
- Derive new PID type from base class
  - Add any datastore specific attributes
- Derive new PDS type from base class
- Specialize the following PDS methods:
  - store
  - restore
  - delete

A Case Study Of DB2

DB2 Specialized Components
DB2 Persistent ID
DB2 Schema Mapper
Sample Client Code
Demo

IBM

DB2 Specialized Components


Client
PO
PID_DB2
Externalization Protocol
POM
PDS_DB2
SchemaMapper
DB2

DB2 Persistent Data Service

- 100% OMG Compliant
- Low Cost, High Performance
- Schema Mapper maps between Object Data and Relational Data
- Part of a family of OMG Services
  - Transaction
  - Concurrency
  - Security
  - Query

DB2 Persistent Identifier

- Uniquely identifies a row
  - datastore type
  - database alias
  - schema mapper (if not using the default)
  - key data (via inherited stream protocol)
- Interface Definition:

```
interface PID_DB2 : CosPersistencePID::PID_DS {
    attribute string dbalias;
    attribute schema_mapper;
    implementation {
      .....
    };
}:
```

Creating & Setting up a PID_DB2

```
...
CosPersistenceDB2_PID_DB2        pidJohn;
CosPersistencePID_PIDFactory     pidf;
CosPersistencePOF_FactoryFinder ff;

ff    = CosPersistence_getGlobalFactoryFinder(ev);
pidf = CosPersistencePOF_FactoryFinder_findFactory
 (ff, ev, "CosPersistenceDB2::PID_DB2", CosPersistencePOF_type

pidJohn = _create(pidf, ev);

__set_datastore_type(pidJohn, ev, "DB2");
__set_dbalias(pidJohn, ev, "IBM");
_write_string(pidJohn, ev, "012345");
```

Object Schema Mapper

- Provides mapping between an object's datastream and DB2 Schema Definition
- Schema Mapper is a base class defining 3 methods: store, restore, delete
- Specialized Schema Mappers are written for each object type
- PDS routes to appropriate Schema Mapper

Graphical Front End Tool

- Creates object definitions
- Creates and drops DB2 tables and views
- Defines mapping between object definition and DB2 schema

Sample Client Code

- **Description:**
  - Manages Employee and Customer Information for Company
- **Storage Facilities:**
  - DB2
  - POSIX file
- **Business Objects:**
  - Company
  - Customer
  - Employee

DB2 Schema Definitions

Employee Table:

Column Name	Data Type	Description
Name	CHAR(100)	Name of the employee
Age	SHORT	Employee's age
Salary	DOUBLE	Employee's salary

Customer Table:

Column Name	Data Type	Description
Name	CHAR(100)	Name of the customer
Age	SHORT	Customer's age
CreditLimit	DOUBLE	Customer's credit limit

Person Definition

```
interface person : PO, Streamable {

  attribute string name;
  attribute short age;

  implementation {
     override: externalize_to_stream,
                  internalize_from_stream;
  };
};
```

Person Implementation

```
void externalize_to_stream(person somSelf,
                           Environment *ev,
                           Stream stream)
{
 personData *somThis = personGetData(somSelf);

  _write_string(stream, ev, _name);
  _write_short(stream, ev, _age);
}
```

Person Implementation (cont.)

```
void internalize_from_stream(person somSelf,
                             Environment *ev,
                             Stream stream)
{
 personData *somThis = personGetData(somSelf);

 _name = _read_string(stream, ev);
 _age = _read_short(stream, ev);
}
```

Customer Definition

```
interface customer : person {
  attribute long creditLimit;
  implementation {
   override: externalize_to_stream,
             internalize_from_stream;
  };
};
```

Customer Implementation

```
void externalize_to_stream(customer somSelf,
                           Environment *ev,
                           Stream stream)
{
 customerData *somThis = customerGetData(somSelf);
 customer_parent_person_externalize_to_stream(somSelf,
                                              ev, stream);

  _write_long(stream, ev, _creditLimit);
}
```

Customer Implementation (cont.)

```
void internalize_from_stream(customer somSelf,
                             Environment ev,
                             Stream stream)
{
 customerData *somThis = customerGetData(somSelf)
 customer_parent_person_internalize_from_stream
                                (somSelf, ev, stream);
  _creditLimit = _read_long(stream, ev);
}
```

Customer Schema Mapper

```
interface customer_sm : SchemaMapper {

  implementation {

    override: store, restore, Delete;
  };
};
```

Customer Schema Mapper

```
#include "cust_sm.ih"
#include <somphdr>
#include <sqlenv.h>
#include <customer.h>

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
    char     Name[100];
    short    Age;
    double   CreditLimit;
EXEC SQL END DECLARE SECTION;
```

Customer Schema Mapper (cont.)

```
SOM_Scope void*  SOMLINK store (customer_sm somSelf,
                             Environment *ev,
                             CosStream_StreamIO objstr,
                             CosPersistenceDB2_PID_DB2 pid)
{
  strcpy(Name, _read_string(objstr, ev));
  Age    = _read_short(objstr, ev);
  CreditLimit = _read_long(objstr, ev);

  EXEC SQL UPDATE CUSTOMER
      SET AGE = :Age, CREDITLIMIT = :CreditLimit
      WHERE NAME = :Name;
  ...
```

Customer Schema Mapper (cont.)

```
....

if (sqlca->sqlcode != SQL_RC_OK) {
     EXEC SQL INSERT INTO CUSTOMER
          VALUES (:Name, :Age, :CreditLimit);
  }

  return (&sqlca);
}
```

Customer Schema Mapper (cont.)

```
SOM_Scope void*  SOMLINK restore (customer_sm somSel
                  Environment *ev,  CosStream_StreamIO objstr,
                   CosPersistenceDB2_PID_DB2 pid)
{
  strcpy(Name, (char *) _read_string(pid, ev));

  EXEC SQL SELECT AGE, CREDITLIMIT
      INTO :Age, :CreditLimit
      FROM  CUSTOMER WHERE NAME = :Name;

  _write_string(objstr, ev, Name);
  _write_short(objstr, ev, Age);
  _write_long(objstr, ev, CreditLimit);

  return (&sqlca);
}
```

Customer Schema Mapper (cont.)

```
SOM_Scope void*  SOMLINK Delete (customer_sm somSel
                        Environment *ev,
                        CosPersistenceDB2_PID_DB2 pid)
{
  strcpy(Name, (char *) _read_string(pid, ev));

  EXEC SQL DELETE FROM CUSTOMER
      WHERE NAME = :Name;

  return (&sqlca);
}
```

Company Definition

```
interface customer;
interface employee;

interface company : PO, Streamable {

  const long MaxCustomers = 20;
  const long MaxEmployees = 20;

  void add_customer(in customer newCustomer);
  void add_employee(in employee newEmployee);
```

Company Definition (cont.)

```
implementation {

  sequence <employee, MaxEmployees>employees;
  sequence <customer, MaxCustomers> customers;

   override:  externalize_to_stream,
              internalize_from_stream;

 };
};
```

Company Implementation

```
SOM_Scope void  SOMLINK add_customer(company
        somSelf,  Environment *ev, customer newCustomer)
{
   companyData *somThis =companyGetData(somSelf);
   _customers._buffer[_customers._length] = newCustomer
   _customers._length++;
}
```

Company Implementation (cont.)

```
SOM_Scope void  SOMLINK add_employee(company
        somSelf, Environment *ev, employee newEmployee)
{
   companyData *somThis = companyGetData(somSelf);
   _employees._buffer[_employees._length]=newEmployee;
   _employees._length++;
}
```

Company Implementation (cont.)

```
SOM_Scope void  SOMLINK externalize_to_stream(
    company somSelf,  Environment *ev, Stream stream)
{
   long n;
   person next;
   companyData *somThis=companyGetData(somSelf);
   _write_long(stream, ev, _customers._length);
   for (n=0; n< _customers._length; n++) {
         next = _customers._buffer[n];
         _write_object(stream, ev, next);
   }
```

Company Implementation (cont.)

```
   _write_long(stream, ev, _employees._length);
   for (n=0; n< _employees._length; n++) {
         next = _employees._buffer[n];
         _write_object(stream, ev, next);
   }
}
```

Company Implementation (cont.)

```
SOM_Scope void  SOMLINK internalize_from_stream(
        company somSelf,  Environment *ev, Stream stream)
{
    long n, npeople;
    person next;
    companyData *somThis=companyGetData(somSelf);

    npeople = _read_long(stream, ev);
    for (n=0; n<npeople; n++) {
      next = _read_object(stream, ev);
      _add_customer(somSelf, ev, next);
    }
```

Company Implementation (cont.)

```
npeople = _write_long(stream, ev);
    for (n=0; n<npeople; n++) {
      next = _write_object(stream, ev);
      _add_employee(somSelf, ev, next);
    }
}
```

Sample Program

```
demo (FILE *input, FILE *output)
{
 Environment  *ev;
 employee      John;
 customer      Anne;
 company       company,

 CosPersistencePOSIX_PID_POSIX  pidCompany;
 CosPersistenceDB2_PID_DB2      pidJohn, pidAnne;
 CosPersistencePO_POFactory     pof_comp;
 CosPersistencePO_POFactory     pof_cust, pof_emp;
 CosPersistencePID_PIDFactory   pidf_db2, pidf_posix;
 CosPersistencePOF_FactoryFinder ff;
```

Sample Program (cont.)

```
ev = SOM_CreateLocalEnvironment();
ff  = CosPersistence_getGlobalFactoryFinder(ev);

pidf_posix  =  CosPersistencePOF_FactoryFinder_findFactory
                     (ff,ev,"CosPersistencePOSIX::PID_POSIX",
                                   CosPersistencePOF_type);
pidf_db2 = CosPersistencePOF_FactoryFinder_findFactory
                     (ff,ev,"CosPersistenceDB2::PID_DB2",
                                   CosPersistencePOF_type);

pof_comp = CosPersistencePOF_FactoryFinder_findFactory
                  (ff,ev, "company", CosPersistencePOF_type);
pof_emp = CosPersistencePOF_FactoryFinder_findFactory
                  (ff,ev, "employee", CosPersistencePOF_type);
pof_cust = CosPersistencePOF_FactoryFinder_findFactory
                  (ff,ev, "customer", CosPersistencePOF_type);
```

Sample Program (cont.)

```
pidCompany =_create(pidf_posix, ev);
__set_datastore_type(pidCompany, ev, "File1");
__set_pathName(pidCompany, ev, "company.dat");
company = _createPO(pof_comp, ev, pidCompany);

pidAnne = _create(pidf_db2, ev);
__set_datastore_type(pidAnne, ev, "DB2");
__set_dbalias(pidAnne, ev, "company");
_write_string(pidAnne, ev, "Anne");

pidJohn = _create(pidf_db2, ev);
__set_datastore_type(pidJohn, ev, "DB2");
__set_dbalias(pidJohn, ev, "company");
_write_string(pidJohn, ev, "John");
```

Sample Program (cont.)

```
John = _createPO(pof_emp, ev, pidJohn);
__set_name(John, ev, "John");
__set_age(John, ev, 22);
__set_salary(John, ev, 38000);
_add_employee(company, ev, John);

Anne = _createPO(pof_cust, ev, pidAnne);
__set_name(Anne, ev, "Anne");
__set_age(Anne, ev, 20);
__set_creditLimit(Anne, ev, 1000);
_add_customer(company, ev, Anne);
```

Sample Program (cont.)

```
   CosPersistencePO_PO_store(company, ev, NULL);
   CosPersistencePO_PO_store(John, ev, NULL);
   CosPersistencePO_PO_store(Anne, ev, NULL);

   _somFree(John);
   _somFree(Anne);
   _somFree(company);

}
```

A Family of OMG Services...

- Transaction Service
  - Allows objects to participate in transactions
- Security Service
  - Provides user access control to objects at the method lev
- Concurrency Service
  - Coordinates multi-user access to object data
- Query Service
  - Finds individual objects within collections

Transaction Service

```
tran = SomGetCurrentTran();

done = FALSE;
while !(done) {
   accntNo = readAccntNumber();
   _write_string(pid, ev, accntNo);
   Transaction_Current_begin(tran, ev);
   _restore(accnt, pid);
   updateAccount(accnt);
   _store(accnt, pid);
   Transaction_Current_commit(tran, ev);
}
```

Concurrency Service

```
done = FALSE;
while !(done) {
  accntNo = readAccntNumber();
  _write_string(pid, ev, accntNo);
  Account_lock(_accnt, Concurrency_write);
  _restore(accnt, pid);
  updateAccount(accnt);
  _store(accnt, pid);
  Account_unlock(_accnt, Concurrency_write);
  _end();
}
```

The Meaning of Datastore Independence

- Same client code works for all datastore
- Binary objects can be redirected to new datastores
  - Even at run time!
- Object Implementor doesn't know what datastore will be used
- Objects can be moved from one datastore to another without change

Demo...

Summary

- Datastore independence
- Industry Standard from the Company that wrote the Standard
- OO Access to Corporate Centric datastores (e.g. DB2/x, IMS, VSAM, CICS)
- Support for wide range of datastores
- Defined mechanisms for datastore vendors to plug in.

The Future of Object Persistence

SYS 0308

Inside OS/2 Warp, a Technical View

Ron Cadima

IBM

Ron Cadima
ISV Development Support
IBM
Boca Raton, FL

ABOUT THIS DOCUMENT

In this paper we will discuss the internals of OS/2 Warp. We will look at tasking and scheduling, semaphore support, and memory management support. The paper is intended for application developers and those who wish to know what is going on inside OS/2 Warp so they can gain some insight on how to best take advantage of it. When OS/2 is used within the paper, it is referring to both the OS/2 Warp for Windows Version 3.0, and the OS/2 Warp Version 3.0 that contains the Win-OS/2 support. This last version is sometimes referred to as OS/2 Fullpack.

It should be noted that the information presented in this paper represents the current implementation of OS/2 and is subject to change in any future release. Continuing changes and improvements are being made to OS/2 Warp to improve it's performance, usability and security. We are constantly looking for better ways to do things and as we find them, they are and will be incorporated into OS/2. Also, many of the algorithms and techniques used within the OS/2 system are covered by copyright laws as well as patented.

Whenever I refer to a DOS application or program in the document, I am referring to programs that run in a DOS environment. This includes all of the Windows programs that are supported.

Trademarks
OS/2 Warp Version 3.0
 IBM Corp.
 Armonk, NY

SCHEDULING AND TASKING

The basic control point within OS/2 is the process. Whenever a program is started, OS/2 creates a process. The process is the entity that owns all of the resources that are used by the program. This includes files, semaphores, global data, printers, etc. All of these resources are used by threads within the process. Even though the thread is the one that opens or creates a file, pipe, semaphore, etc., it is owned by the process. If the thread does not explicitly close the resource before it terminates, the resource will remain active and us up resources until the process terminates. The Thread is the basic execution level of the process or program. When a process is scheduled to run, that is when OS/2 gives it machine cycles to process it,s code instructions, it does so at a thread level. The process can have one or more threads associated with it. Each of these threads will be scheduled individually, allowing a process to be doing multiple things at the same time. It can be recalculating data in a spread sheet at the same time it is formatting a document and reading data from the disk. Even through these threads can process independently of each other, they all are identified within the context of the process and when the state of the process changes, the state of each thread also changes.

PROCESS STATES

Processes can be either in a Foreground State, or a Background State. The threads in the process will have their priority modified depending on which state the process is in. In the Foreground State, the program is the one which has the focus on the screen. It is the highlighted application and the one that the user is interacting with. Since this is the application that the user is currently interacting with, all the threads within the process get a priority boost so they will process before threads in other processes that the user is not currently interacting with. These other threads will be in what is called Background state. They are not directly interacting with the user, and therefore run at whatever priority the application developer assigned to them, or at the default priority if the developer did not assign any specific priority.

THREADS

Let's take a closer look at threads. As we said, the thread is the execution level of a process. Each thread is scheduled to receive what is called a time slice in which to process it's code instructions. The scheduling of these threads is done based on the priority of the thread. The state of the thread will affect its priority and determines when it will run. Threads can be in many different states. Lets first look at those states that do not effect the priority of the thread.

READY

In this state, the thread has instructions it can execute. It is waiting for a time slice to be given to it by the scheduler. While the thread is waiting, it is placed in a thread dispatch queue based on its priority relative to the priority of other threads in the Ready state.

ACTIVE

When in this state, the thread is executing its instruction. Only one thread can be in this state at any given time.

SLEEPING

A thread in this state is waiting for something to happen within a given amount of time. The thread could have issued a command to wait for 30 seconds before continuing to process, or it could be waiting on some other event, such as a semaphore to be posted or cleared, and wants to know if this has not happened after a certain period of time.

FROZEN

Threads in this state are waiting indefinitely for something to happen. The thread is blocked from executing until some specific event or events take place. A thread is also placed in this state when it cannot be dispatched. This is usually caused by a thread in a process issuing a DOSEnterCritSit command. When this is done, all other threads in the same process will enter a Frozen state.

Now let's look at the states which will cause the priority of the thread to be changed. Even though these states are set for Idle class threads, the priority of the idle class thread will not change. It will always run at priority level 100. It is noted in the state description as to whether DOS applications will be effected or not.

FOREGROUND

The thread that has screen focus will receive a boost in its priority. The priority boost is not given to a DOS thread. When the thread no longer has foreground focus, it loses the priority boost.

STARVED

A thread is starved when it has been in the ready state for the amount of time specified in the MAXWAIT parameter in the CONFIG.SYS file, and if it has not received a time slice in which to run. When in this state, the thread will receive a boost in it's priority to try to give it a time slice to run in. After it has received a single time slice, it's priority will be set back to whatever it normally would be. If the thread gets preempted by a higher priority thread during its time slice, it will lose its priority boost. DOS applications do receive this boost.

IO_CONTROL

In this state, the foreground thread is ready to start processing after the completion of an I/O operation. The threads priority will be boosted for the first time slice it receives after the I/O completed. After the first time slice, the priority boost will be removed. This is done to alleviate any problems that may arise in disk contention, and buffer and cache integrity. Also, this insures that the application a user is interacting with gets the highest priority possible to insure good response times. The DOS thread will receive this priority boost.

KEYBOARD FOCUS

The thread which is interacting directly with the keyboard will receive a priority boost. Since this directly affects the performance that the user perceives, we want to insure that we give priority to threads that the user is interacting with. The DOS applications will receive this boost every time a key is entered on the keyboard. This boost only applies to the process of getting the character into the buffer. Once that is done, the DOS thread returns to normal priority.

WINDOWED

Whenever a thread is trying to write to a window on the screen, it will receive a priority a boost to try an get that done as fast and transparent as possible. A DOS application does not receive this boost.

It is possible for an OS/2 thread to be in more than one of the above states. If it is, the states will be combined to calculate what the priority of the thread should be. In the case of DOS threads, the states of the thread are not combined. Whichever is the last state to be entered will determine the priority of the DOS thread.

PRIORITY LEVEL AND CLASS

The thread states just discussed are used in conjunction with the priority level and class that are assigned to a thread to determine its priority. The level and class are assigned by the developer or left to the system default.

There are currently four priority levels within OS/2. They are:

NORMAL

Which is the default when no priority is assigned. The priority level associated with this is 200. This is also the priority level at which all DOS applications run.

SERVER

The server class is typically used by database and communication applications. It's base priority level is 300. The server class is used for special applications that need to have a higher priority than normal applications because they perform a lot of transaction processing or serve the needs of many other processes and threads.

TIME CRITICAL

Time critical threads have the highest priority level, 800, of all the threads. This level is used for threads that must insure their processing gets done as soon as possible. These threads are used for process control type operations, Communications to insure that lines remain active, and by the system for such functions as lazy writing from the cache, page aging in over committed situations, and others where the integrity of the system needs to be maintained.

IDLE

Idle threads are assigned a priority level of 100. These are threads which do not have to run until the system is not doing anything. These threads do not receive any priority boosts and run the risk of not getting any time slices in which to process. This is especially true when DOS applications are running in the system because they tend to do hardware polling and are therefore always ready to run.

Associated with each priority level is a priority class. The class is used to modify the priority level to give a thread more priority than others at the same level. Priority classes range from a low of 0 to a high of 31. The class applies to all priority levels and types of processes, including DOS applications. Care must be taken when setting priority levels and classes to insure that other threads do not get starved or locked out from processing. This is particularly true with DOS applications. Setting the class at a higher value for one DOS process could starve all the other DOS processes in the machine or any OS/2 process that is running at normal (200) priority.

The table bellow shows the relative priority boosts for different thread states. It just shows the priority for a single state and is provided to give you an idea of what they are.You can see the state of a thread by using the kernel debugger to look at the TCB associated with the thread. By setting a break point at _TKUpdateProirity, you can see the priority that is calculated for the given state.

Thread	Thread Level		
State	Regular	Server	Time Critical
Starved	0x71F	0x72F	0x800
IO Complete	0x72F	0x73F	0x81F
Keyboard Control	0x500	0x51F	0x800
Windowed	0x500	0x500	0x800
Foreground Focus	0x300	0x51f	0x800
Background	0x200	0x300	0x800

Once again, these numbers are subject to change with subsequent changes and releases of OS/2.

SCHEDULING

Now let's look at how threads get scheduled. As threads become ready to run, a kernel API call is made to update their state. Scheduler code looks at the state the thread is currently in and at the new state it wishes to be in and verifies that the states are correct. If the states are correct, then the current state will be modified and the thread will be placed into a queue based on it's state and priority. If the new state is Ready, the scheduler compares the priority of the Ready thread to the priority of the Active thread. If the new thread has a higher priority, the Active thread will be preempted, placed back onto the scheduler queue, and the new thread will receive it's time slice. If the priority is the same or lower than the active thread, the new thread will be placed in the ready queue based on it's priority. When the active threads time slice is used up, the scheduler will look at the state of the active thread. If it still wants to run, and it has a higher priority than the thread at the top of the ready queue, it will receive another time slice and continue to run. If it's priority is equal to or less than the thread that is at the top of the ready queue, it's state will be changed to Ready, it will be placed in the ready queue, and the thread at the top of the ready will change to Active state and receive a time slice in which to run.

As threads become ready because a timeout value has expired or a semaphore has been cleared, their states will be updated and they will go through the scheduler logic described above. In the current implementation of OS/2, when multiple threads become ready because of a semaphore being posted or cleared, the highest priority thread will go through the above process and all other threads will be placed in a delay queue, with a state of Delayed. The delay queue will be processed in subsequent passes through the scheduler.

New in Warp are some additional scheduling changes that allow a thread to get an extra time slice even if the next thread that is ready to run is at the same priority. If the active thread took a page fault, or had a cache hit while doing disk I/O, it will be given an additional time slice. A flag will be set in the TCB to indicate that it has received this additional time slice so the system can check it and insure that it does not receive two consecutive additional time slices.

A thread receives a minimum time slice of approximately 32 milliseconds. This can be dynamically changed by the system to a higher value to help alleviate problems associated with memory over commit. That is when you need more memory to run than is physically in your machine and the system has to swap data and code from memory to disk and back. A thread will not receive it's full time slice if it is preempted by a higher priority thread or if it performs a yield, either directly via an API call, or indirectly by waiting on an event such as a semaphore or some type of I/O.

Threads that have a level of Time Critical are handle through a special dispatch queue which is tuned for performance. They also go through a tuned code path within the scheduler code but are processed basically in the same way as described above within a time critical processing context.

DOS priority scheduling is handled by the DOS VDD (Virtual Device Drivers). The VDDs will issue calls to the OS/2 kernel via DOS helper routines to change the priority of the DOS thread when warranted. This done for keyboard and mouse input, reset to normal priority, and I/O boosts. For interrupt priority boosts, a call is made directly to the scheduler routine that calculates the threads priority. For the disk I/O complete boost, this is handled through the normal scheduler state checking code

If you are using the kernel debugger or PSTAT to look at the priority of blocked threads, remember the thread priority you see is priority of the thread when it blocked. Its priority will be recalculated when it changes to a Ready state. Also, if you are using PSTAT to look at thread priorities, the PSTAT thread has focus and therefore the priority boost applied to it. Since no keyboard activity is taking place while the data is being displayed, you will not see this priority boost.

SEMAPHORES

Now let us turn our attention to semaphores. OS/2 support up to 64,000 system semaphores and 64.000 private semaphores. The private semaphores are on a per process basis. System semaphores are what is called shared semaphores. These are accessible by any thread in the system, and private semaphores are only accessible by all the threads within a process, provided they know about the semaphore. The three types of semaphores are:

MUTEX

This is the mutual exclusive use semaphore. Basically this is used to serialize access to a memory object or some resource. A thread issues a request for the semaphore. When control is received back, if no error or time out has occurred, the thread has control of the resource. It keeps control until it releases the semaphore.

This is the fastest of all the semaphores. The checking of the semaphore state is done at Ring 3. A transition to kernel mode will only be done when the requested semaphore is owned and the requesting thread has to block. Also, when releasing the semaphore, a transition to kernel mode will only be done if a thread is waiting to acquire the semaphore, otherwise, the state is updated at the Ring 3 level.

EVENT

An Event semaphore is used to wait until some event has taken place. This could be waiting for disk I/O, waiting for another thread to process, or any other type of event. This is the second fastest type of semaphore in terms of processing speed. All requests require a ring transition into kernel mode in order to process the request.

MUXWAIT

The MuxWait semaphore allows a thread to wait on multiple Event semaphores, or multiple Mutex semaphores. They can not be combined. This is the slowest of all semaphores since it involves processing a tree of multiple control blocks within multiple threads. All the processing is done while in kernel mode.

In the current version of OS/2 there has been a change in the way semaphores are being handled. When a request is issued for a semaphore with a timeout value of zero, OS/2 will now do a yield to allow any other ready thread a chance to run. In the previous versions of OS/2, control is returned back to the calling thread with no yield being done and it would not allow another thread to get serviced. The change helps performance when multiple threads are ready to run.

There has also been a change in how the control blocks for shared semaphores are being allocated. Now they are being allocated from a kernel heap. Also, there are initially 112 control blocks allocated for shared semaphores. This is enough for the base system and 2 or 3 applications. When the number of shared semaphore control blocks needs to be increased, we now grow the size of the table by 8 new control blocks at a time. In the version before OS/2 Warp, 32 control blocks were initially allocated and the number of shared semaphore control blocks would be doubled every time the table had to be expanded. Since at system initialization, were were using 52 semaphores, the table had to grow immediately. When the first applications start, the table was grow again by doubling its size. This lead to inefficient use of space where many semaphore control blocks would go unused in the system. With the current implementation, there is a performance impact on test programs that just go out and allocate shared as many semaphores as they can. This is because the semaphore table has to grow a lot more often for this type of test than it had to previously. But, in real applications, there has been no slow down and more memory has been saved.

SEMAPHORE DEBUGGING

Semaphores are one of the major causes of programs hangs that exist in OS/2. The OS/2 kernel debugger can be used to look at the control blocks for shared semaphores and determine who created the semaphore, the state of the semaphore and how many threads are waiting on the semaphore. You can also use the PSTAT utility to determine how many reference exists for system semaphores and how many requests are outstanding. _PShSemTbl is the name of the shared semaphore table. This table contains a list of pointers to the actual semaphore control blocks. The last half of the semaphore handle is the index into this table. _ulShTotUsed contains the number of semaphores that are currently in use, and _ulShTblSize is the total number that have been allocated.

The handle associated with the semaphore also relates to what type it is. If the high order bit is on, then it is a shared semaphore. If bit 0x0001000 is on, then it is a 32 bit semaphore. If bit 0x0010 is set on, then it is event semaphore. Within the semaphore control block itself, at an offset of 20 bytes into the table is a field that contains the address of the control block for the thread that created the semaphore. This field only exists in the debug version of the OS/2 kernel. The first word in the table contains the flag bytes used for the semaphore control block. A value of 0x0040 means that the semaphore has been posted. 0x0080 means the owning process has died, and 0x0100 means that the MuxWait semaphore contains MUTEX semaphores in it. Also, if you look 8 Bytes into the Mutex semaphores control block, you will see the slot number of the thread that owns the semaphore. As warned earlier, these values are only valid for the current release of OS/2 Warp Version 3 and are subject to change.

MEMORY MANAGEMENT

Now lets look at memory management within OS/2 Warp. We start with page aging.

PAGE AGING

Page aging is done within a separate thread that runs at different priorities based on the amount of memory over commit that exists. It will run when a threshold has been reached based on the number of idle pages in the system. Idle pages are defined as those pages that are free, not accessed and discardable or swappable. When the idle list threshold is reached, the ager thread will be scheduled to run and all the threads that are not active, will be scanned for pages that have not been accessed since the last time the ager thread ran. The scan is done on a process basis. Those pages that are associated with kernel code and data, or the currently active process or not scanned. The scanning of page table entries is done until enough pages have been found, no more processes can be scanned, or too much time has expired. If not enough pages have been found, then we will look at pages owned by the kernel as candidates for idling.

2Page Swapping

In OS/2, we perform demand page swapping. When a page is needed for a program that is being loaded or memory that is being allocated, a search is made of the idle list to find a page that is free or discardable. If one is found, then that page will be used to satisfy the page needed. If not, we attempt to gather eight consecutive pages together and swap those out to the swap file. Those 8 pages are marked present and swapped, so that they can be reclaimed if they are accessed again before they are used for another page. The swapping of pages to disk is via a fast path through the file system, and the BUFFERS defined in the CONFIG.SYS file are used as a staging area for the disk I/O so the disk cache will not be cluttered with page data. If the page table needs to grow, it will grow in 512K increments. When pages need to brought back into memory, this is done one page at a time at the time of the page fault.

MEMORY ALLOCATIONS

Memory allocations cause page table entries to be created for the specific object that is being created. A minimum of 64K of memory will be mapped into a page table in order to support tiling of 16 Bit applications that use the segmented memory model. Physical memory for the pages is allocated at the time the page is actually or touched, or if the allocating API or CONFIG.SYS statement, specify the memory is to be committed, then physical and/or swapper memory will be allocated for the memory object. In either case, the first time the page is accessed, it will be zero filled by the system.

Memory can also be allocated as contiguous memory. This is usually done for buffers that will be used to store data for use by older DMA controllers that require contiguous memory. This becomes a very expensive type of memory allocation because it usually causes memory to be moved in RAM since large runs of free pages are not generally available in the system. This movement takes place at the time the object is allocated. If contiguous memory is required, it should be allocated once and reused for the entire life of the program.

Memory locking can also be an expensive operation from a performance point of view. From a time perspective, it is more effective to allocate a long term locked piece of memory and reuse it than it is to perform short term locks and unlocks. Care should also be taken to insure that memory that is already locked is not being locked again. This has no effect on the system other than to waste time. Tools such as Theseus2 and OS2MEMU can be used to look at how memory is being allocated and used. These will also tell you who is using the memory and what type of memory it is.

CONCLUSION

In conclusion, the import things to remember are that OS/2 is a preemptive multitasking operating system. It allows you to set the priority of your threads based on relative importance of the processing they are doing. However, when setting priorities, be aware of other threads and processes that may be running in the system and that your thread may not be as important or more import than those other threads. It is usually better to let the priorities take the system defaults and let the OS/2 system change the priorities based on what the user is doing on the system.

When using semaphores, Mutex semaphores offer the best performance. Be careful in using semaphores that you do not get yourself into a lockout situation. This typically happens when a thread is waiting for a semaphore to be cleared by another thread that is blocked.

For memory allocations, it is better to allocate memory as uncommitted, and then commit that memory as you need it. Always preallocate all the memory you need and reuse memory instead of allocating and deallocating it. This is also true for locked memory. Do a long term lock on memory and reuse it instead of doing many short term locks and unlocks.

Finally, be aware of what the system is doing and take advantage of what it has to offer.

this page intentionally left blank

CS 0404

DB2 Family: Version 2 of DB2/2 and DB2/6000

Dan Britton

IBM

IBM

DB2 Family Update

DB2 for OS/2 and AIX Version 2

DB2 family

Dan S. Britton
DB2 Family Specialist
IBM Santa Teresa Laboratory
BRITTON@VNET.IBM.COM
512-823-8937

IBM

DB2 for OS/2 & AIX

Disclaimer

The information in this presentation concerns new products that IBM has announced. Any discussion of OEM products is based upon information which has been publicly available and is subject to change. The specification of some of the features described in this presentation may change before the General Availability date of these products.

SPECIAL NOTICES

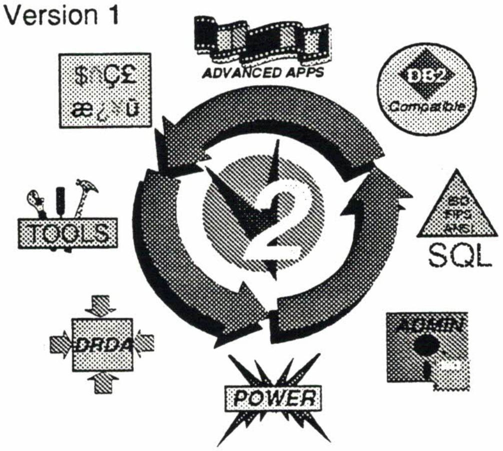
REFERENCES IN THIS PUBLICATION TO IBM PRODUCTS, PROGRAMS, OR SERVICES DO NOT IMPLY THAT IBM INTENDS TO MAKE THESE AVAILABLE IN ALL COUNTRIES IN WHICH IBM OPERATES.

IBM MAY HAVE PATENTS OR PENDING PATENT APPLICATIONS COVERING SUBJECT MATTER IN THIS DOCUMENT. THE FURNISHING OF THIS DOCUMENT DOES NOT IMPLY GIVING LICENSE TO THESE PATENTS.

© Copyright IBM 1995

DB2 for OS/2 and AIX Version 2

- Product features aligned between AIX and OS/2
- Extended relational functionality
- Flexible management of large volumes of data
- Improved interoperability through distributed support
- High performance OLTP and decision-support
- Increased DB2/MVS compatibility
- Upward compatibility from Version 1



© Copyright IBM 1995

Support for Advanced Applications

New Table Constraints

- Constraints on columns (domain support)
- Table level constraints

AlbumNum	Artist	Cover	Media
2435	Bon Happy		CD, Tape

Advanced SQL

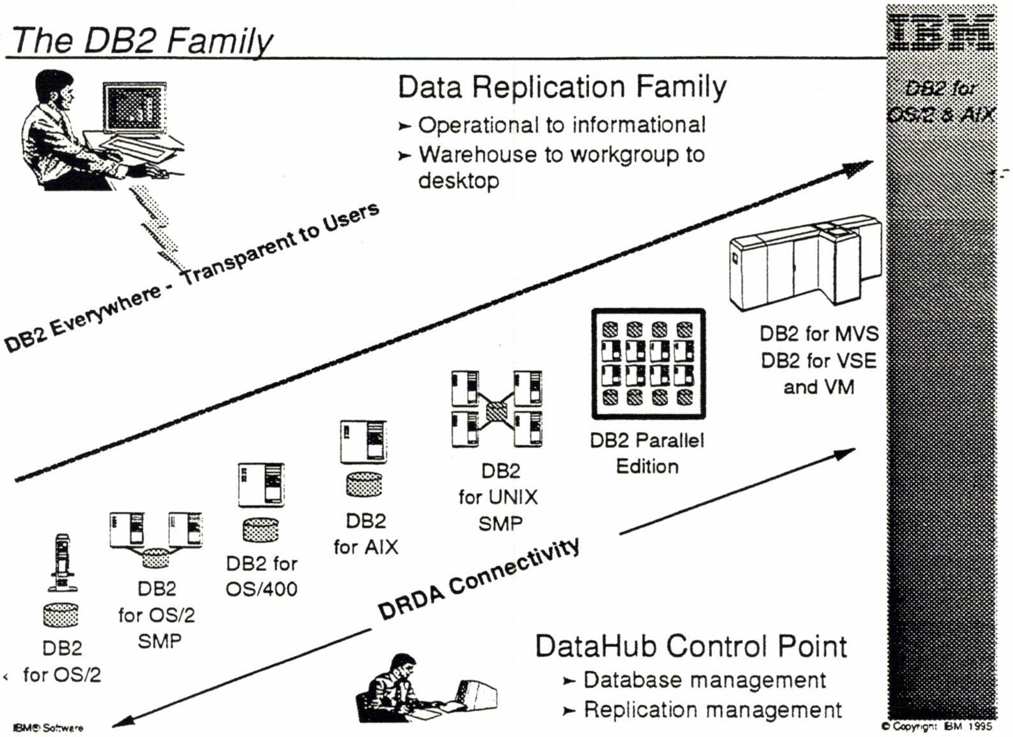
- User-defined functions
- Recursive SQL
- Triggers
- Inline views, nested tables
- SQL'92 and SQL 3 functions

New Datatypes

- Character large objects
- Binary large objects
- User-defined datatypes

© Copyright IBM 1995

The DB2 Family



DB2 for OS/2 & AIX

Data Replication Family

- Operational to informational
- Warehouse to workgroup to desktop

DB2 Everywhere - Transparent to Users

DB2 for OS/2

DB2 for OS/2 SMP

DB2 for OS/400

DB2 for AIX

DB2 for UNIX SMP

DB2 Parallel Edition

DB2 for MVS
DB2 for VSE
and VM

DRDA Connectivity

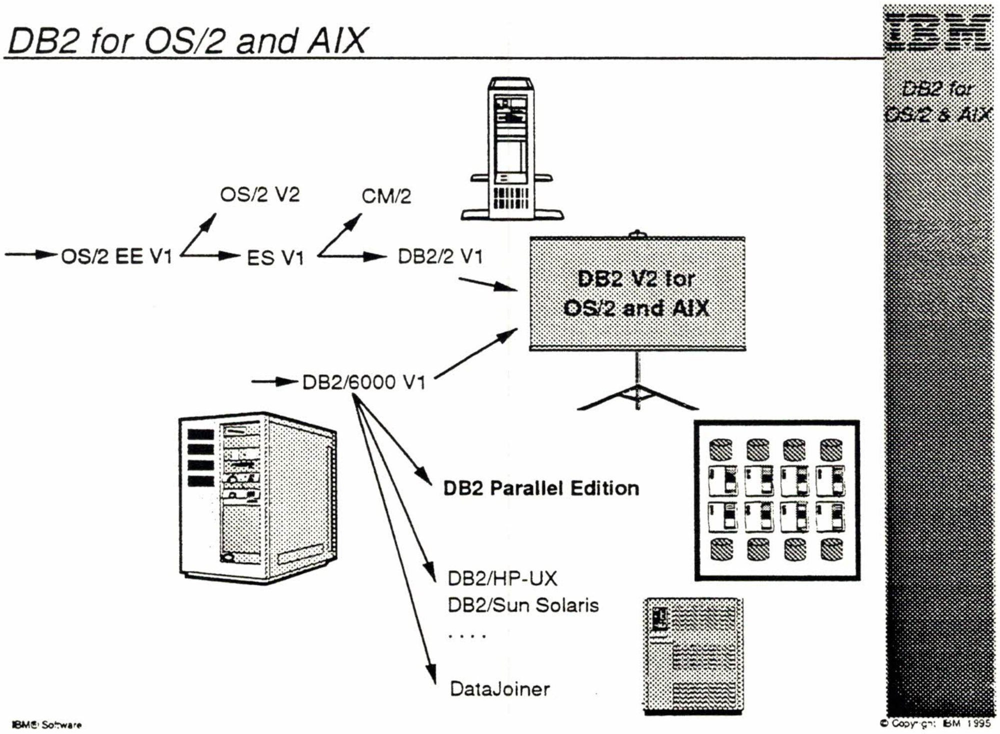
DataHub Control Point

- Database management
- Replication management

IBM® Software

© Copyright IBM 1995

DB2 for OS/2 and AIX



DB2 for OS/2 & AIX

OS/2 EE V1 → OS/2 V2 → ES V1 → CM/2 → DB2/2 V1 → DB2 V2 for OS/2 and AIX

DB2/6000 V1 → DB2 V2 for OS/2 and AIX

DB2/6000 V1 →

DB2 Parallel Edition

DB2/HP-UX
DB2/Sun Solaris
. . . .

DataJoiner

IBM® Software

© Copyright IBM 1995

Table Constraints

DB2 for OS/2 & AIX

Name	Serial	Age	Salary	Department
George	2433	34	20000	133
Fred	2211	23	25000	165
Bill	2411	53	31200	218
Heather	1442	24	18500	133

Datatype Constraints

VARCHAR	SMALLINT	SMALLINT	DECIMAL	CHAR
30			7,2	3
NOT NULL	NOT NULL			

Referential Integrity

	PRIMARY			
	KEY			

Column Constraints

DEPARTMENT BETWEEN 10 AND 200
AGE > 18 AND AGE < 70

Table Constraints

AGE < 25 OR SALARY > 30000
Anyone over 25 must make more than 30000 per year

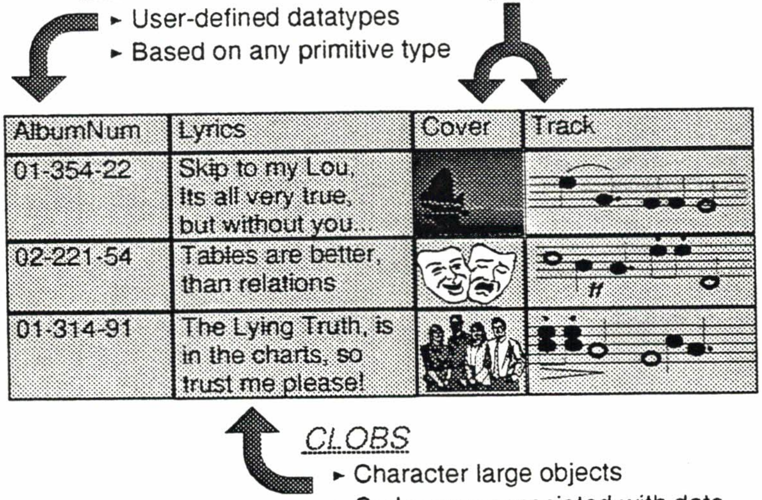
AGE	SALARY	T/F
30	32000	TRUE
30	18000	FALSE
19	32000	TRUE
19	18000	TRUE

IBM® Software

© Copyright IBM 1995

New Datatypes

DB2 for OS/2 & AIX



BLOBS

- Binary large objects
- Any type of data or bit string

UDT

- User-defined datatypes
- Based on any primitive type

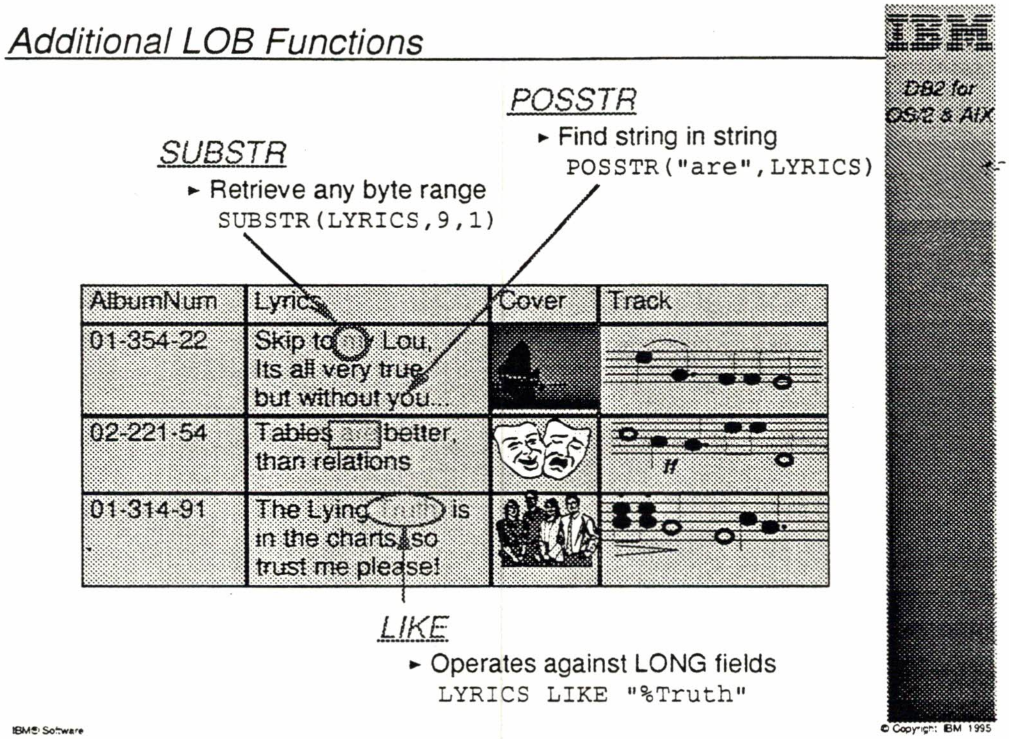
AlbumNum	Lyrics	Cover	Track
01-354-22	Skip to my Lou, Its all very true, but without you...		
02-221-54	Tables are better, than relations		
01-314-91	The Lying Truth, is in the charts, so trust me please!		

CLOBS

- Character large objects
- Code page associated with data
- Automatic conversion

IBM® Software

© Copyright IBM 1995



Additional LOB Functions

DB2 for OS/2 & AIX

POSSTR

- Find string in string
  `POSSTR("are",LYRICS)`

SUBSTR

- Retrieve any byte range
  `SUBSTR(LYRICS,9,1)`

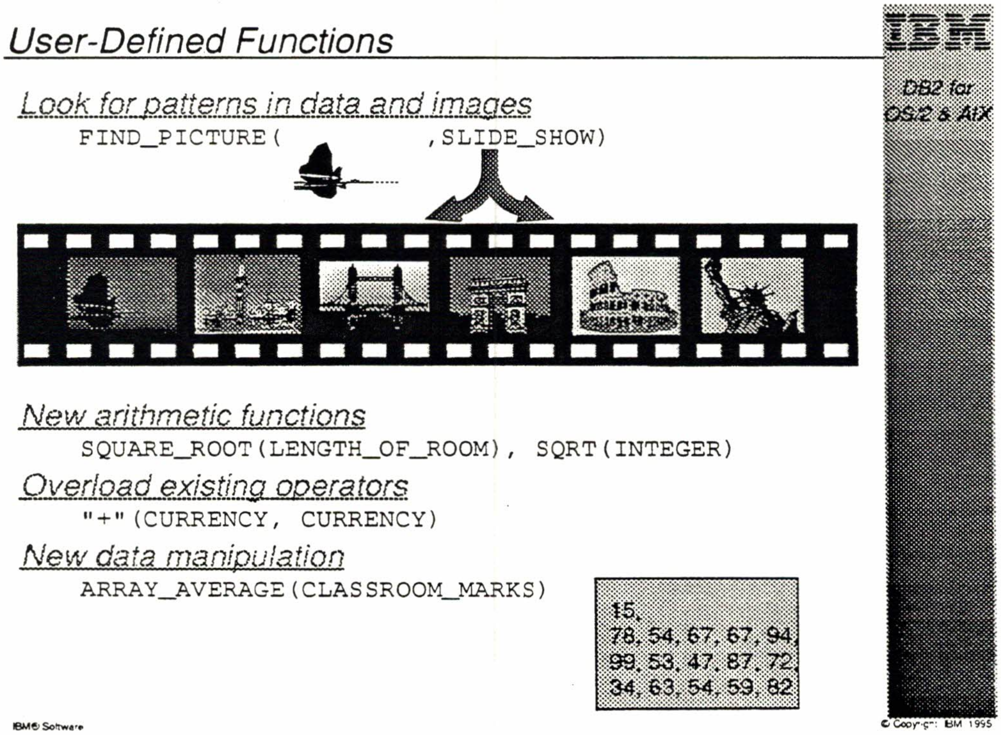
AlbumNum	Lyrics	Cover	Track
01-354-22	Skip to [illegible]y Lou, Its all very true but without you...		
02-221-54	Tables [illegible] better, than relations		
01-314-91	The Lying [illegible] is in the charts so trust me please!		

LIKE

- Operates against LONG fields
  `LYRICS LIKE "%Truth"`

IBM® Software

© Copyright IBM 1995



User-Defined Functions

DB2 for OS/2 & AIX

Look for patterns in data and images

`FIND_PICTURE(` `,SLIDE_SHOW)`

New arithmetic functions

`SQUARE_ROOT(LENGTH_OF_ROOM), SQRT(INTEGER)`

Overload existing operators

`"+"(CURRENCY, CURRENCY)`

New data manipulation

`ARRAY_AVERAGE(CLASSROOM_MARKS)`

15,
78, 54, 67, 67, 94,
99, 53, 47, 87, 72,
34, 63, 54, 59, 82

IBM® Software

© Copyright IBM 1995

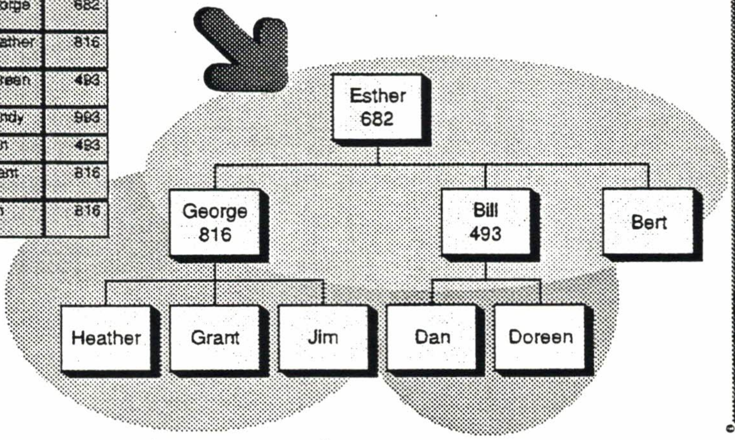
Recursive SQL

DB2 for OS/2 & AIX

Who works in Department 682, either directly or indirectly?

Empno	Name	Dept
1423	Bill	682
2232	John	440
1645	Bert	682
5644	Esther	231
4311	Men	123
7546	George	682
3544	Heather	816
2412	Doreen	493
8673	Randy	993
1422	Dan	493
3533	Grant	816
1522	Jim	816

Dept	MgrNo
682	5644
493	1423
816	7546



IBM® Software

© Copyright IBM 1995

Recursive SQL

DB2 for OS/2 & AIX

DEPARTMENT

Deptno	MgrNo
682	5644
493	1423
816	7548

EMPLOYEE

Serial	Name	Dept
1423	Bill	682
2232	John	440
1645	Bert	682
5644	Esther	231
4311	Men	123
7546	George	682
3544	Heather	816
2412	Doreen	493
8673	Randy	993
1422	Dan	493
3533	Grant	816
1522	Jim	816

```
WITH RECEMP (EMPNO,NAME, DEPT,  MANAGER)
   AS   (SELECT E.EMPNO, E.NAME, E.DEPT,  MGR.NAME
             FROM EMPLOYEE MGR, DEPARTMENT D,
                  EMPLOYEE E
                  WHERE E.DEPT = '682'
                     AND MGR.EMPNO = D.MGRNO
                     AND D.DEPTNO = E.DEPT
        UNION ALL
        SELECT E.EMPNO, E.NAME, E.DEPT, MGR.NAME
             FROM RECEMP MGR, DEPARTMENT D,
                  EMPLOYEE E
                  WHERE MGR.EMPNO = D.MGRNO
                     AND D.DEPT = E.DEPT
        )
SELECT * FROM RECEMP
   ORDER BY DEPT, EMPNO
```

RECEMP

Empno	Name	Dept	Manager
7545	George	682	Esther
1123	Bill	682	Esther
1645	Bert	682	Esther
3544	Heather	816	George
3533	Grant	816	George
1522	Jim	816	George
1422	Dan	493	Bill
2412	Doreen	493	Bill

IBM® Software

© Copyright IBM 1995

Nested Table Expressions

DB2 for OS/2 & AIX

DEPARTMENT

Serial	Salary	Dept
1422	15000	682
2232	18000	440
1645	34000	682
5644	12000	682
4311	32000	123
7546	15600	682
8544	57300	816
2412	34000	482
8673	25400	234
1422	17600	482
3538	64000	816
1522	33000	816

AVG_DEPT_SAL

Dept	AVG_SAL
682	32500
440	19800
123	46980
816	33875
193	17599
234	23850

Common Table Expression

```
WITH AVG_DEPT_SAL(DEPT, AVG_SAL)
    AS (SELECT DEPT, AVG(SALARY)
             FROM DEPARTMENT
             GROUP BY DEPT
        )
SELECT AVG(AVG_SAL)
       FROM AVG_DEPT_SAL
```

Nested Table Expression/Inline View

```
SELECT AVG(AVG_SAL)
     FROM (
          SELECT DEPT,
               AVG(SALARY) AS AVG_SAL
               FROM DEPARTMENT
               GROUP BY DEPT
          ) AS AVG_DEPT_SAL
```

IBM® Software

© Copyright IBM 1995

Triggers

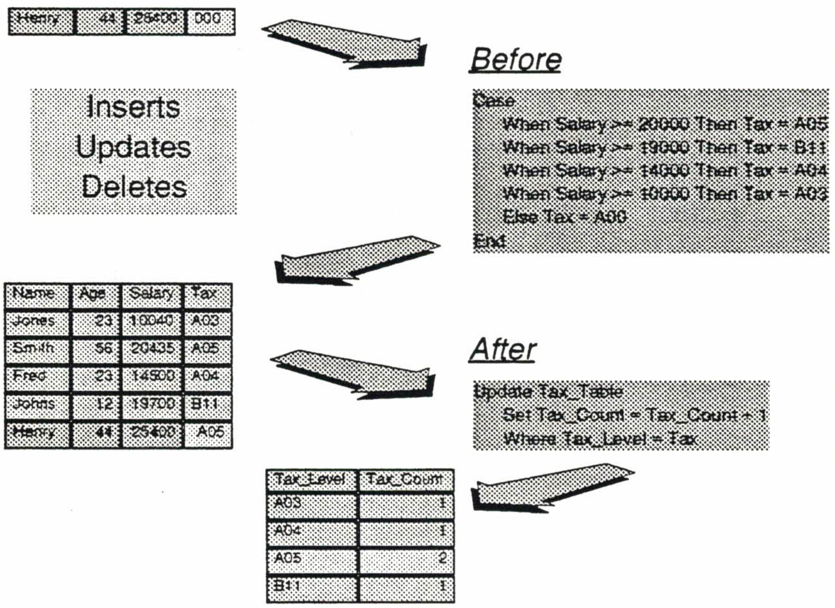
DB2 for OS/2 & AIX

- A Trigger defines a set of actions that are executed at, or "triggered" by, an update operation on a specified base table
- Triggers may be used to:
  - perform validation of input data
  - automatically generate a value for a newly inserted row
  - read from other tables for cross-referencing purposes
  - write to other tables for audit-trail purposes
  - support alerts (through electronic mail messages)
- Triggers result in:
  - faster application development
  - global enforcement of business rules
  - easier maintenance
- CHECK and REFERENTIAL INTEGRITY should be used for declarative constraints

IBM® Software

© Copyright IBM 1995

Triggers



Henry	44	25400	000

Inserts
Updates
Deletes

Before

```
Case
    When Salary >= 20000 Then Tax = A05
    When Salary >= 19000 Then Tax = B11
    When Salary >= 14000 Then Tax = A04
    When Salary >= 10000 Then Tax = A03
    Else Tax = A00
End
```

Name	Age	Salary	Tax
Jones	23	10040	A03
Smith	56	20435	A05
Fred	23	14500	A04
Johns	12	19700	B11
Henry	44	25400	A05

After

```
Update Tax_Table
    Set Tax_Count = Tax_Count + 1
    Where Tax_Level = Tax
```

Tax_Level	Tax_Count
A03	1
A04	1
A05	2
B11	1

© Copyright IBM 1995

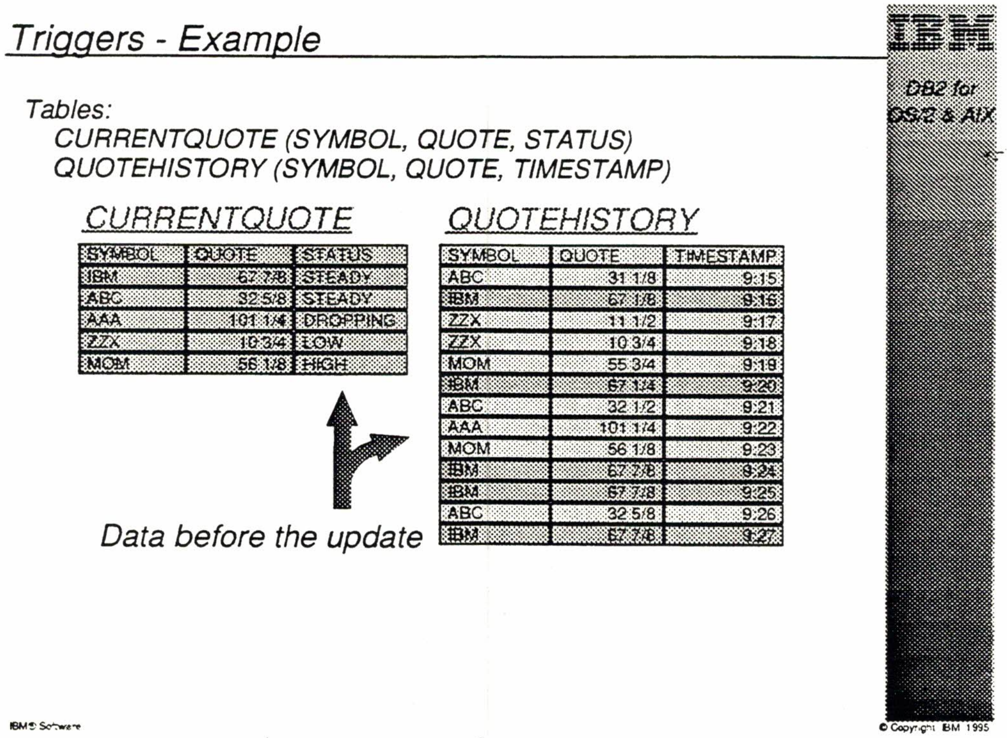
Triggers - Example

EXAMPLE
The following example illustrates the use of a trigger. Consider an application which records and tracks changes to stock prices. The database contains two tables, CURRENTQUOTE and QUOTEHISTORY. When the QUOTE column of CURRENTQUOTE is updated, the new quote should be copied, with a timestamp, to the QUOTEHISTORY table. Also, the STATUS column of CURRENTQUOTE should be updated to reflect whether the stock is:

1. *rising in value;*
2. *at a new high for the year;*
3. *dropping in value;*
4. *at a new low for the year;*
5. *steady in value.*

TABLES:
CURRENTQUOTE (SYMBOL, QUOTE, STATUS)
QUOTEHISTORY (SYMBOL, TIMESTAMP, QUOTE)

© Copyright IBM 1995



Triggers - Example

DB2 for OS/2 & AIX

Tables:
CURRENTQUOTE (SYMBOL, QUOTE, STATUS)
QUOTEHISTORY (SYMBOL, QUOTE, TIMESTAMP)

CURRENTQUOTE

SYMBOL	QUOTE	STATUS
IBM	67 7/8	STEADY
ABC	32 5/8	STEADY
AAA	101 1/4	DROPPING
ZZX	10 3/4	LOW
MOM	56 1/8	HIGH

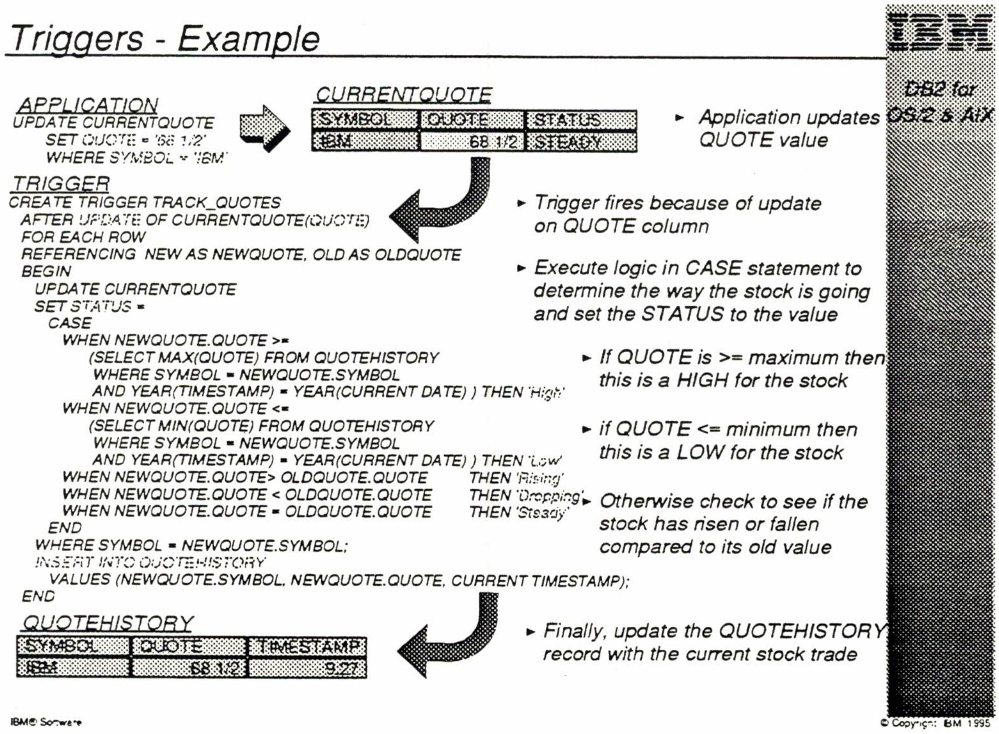
QUOTEHISTORY

SYMBOL	QUOTE	TIMESTAMP
ABC	31 1/8	9:15
IBM	67 1/8	9:16
ZZX	11 1/2	9:17
ZZX	10 3/4	9:18
MOM	55 3/4	9:19
IBM	67 1/4	9:20
ABC	32 1/2	9:21
AAA	101 1/4	9:22
MOM	56 1/8	9:23
IBM	67 7/8	9:24
IBM	67 7/8	9:25
ABC	32 5/8	9:26
IBM	67 7/8	9:27

Data before the update

IBM® Software

© Copyright IBM 1995



Triggers - Example

DB2 for OS/2 & AIX

APPLICATION

```
UPDATE CURRENTQUOTE
    SET QUOTE = '68 1/2'
    WHERE SYMBOL = 'IBM'
```

CURRENTQUOTE

SYMBOL	QUOTE	STATUS
IBM	68 1/2	STEADY

- Application updates QUOTE value

TRIGGER

```
CREATE TRIGGER TRACK_QUOTES
 AFTER UPDATE OF CURRENTQUOTE(QUOTE)
 FOR EACH ROW
 REFERENCING  NEW AS NEWQUOTE, OLD AS OLDQUOTE
 BEGIN
   UPDATE CURRENTQUOTE
   SET STATUS =
     CASE
       WHEN NEWQUOTE.QUOTE >=
           (SELECT MAX(QUOTE) FROM QUOTEHISTORY
            WHERE SYMBOL = NEWQUOTE.SYMBOL
            AND YEAR(TIMESTAMP) = YEAR(CURRENT DATE) ) THEN 'High'
       WHEN NEWQUOTE.QUOTE <=
           (SELECT MIN(QUOTE) FROM QUOTEHISTORY
            WHERE SYMBOL = NEWQUOTE.SYMBOL
            AND YEAR(TIMESTAMP) = YEAR(CURRENT DATE) ) THEN 'Low'
       WHEN NEWQUOTE.QUOTE> OLDQUOTE.QUOTE       THEN 'Rising'
       WHEN NEWQUOTE.QUOTE < OLDQUOTE.QUOTE      THEN 'Dropping'
       WHEN NEWQUOTE.QUOTE = OLDQUOTE.QUOTE      THEN 'Steady'
     END
   WHERE SYMBOL = NEWQUOTE.SYMBOL;
   INSERT INTO QUOTEHISTORY
     VALUES (NEWQUOTE.SYMBOL, NEWQUOTE.QUOTE, CURRENT TIMESTAMP);
 END
```

- Trigger fires because of update on QUOTE column
- Execute logic in CASE statement to determine the way the stock is going and set the STATUS to the value
  - If QUOTE is >= maximum then this is a HIGH for the stock
  - if QUOTE <= minimum then this is a LOW for the stock
  - Otherwise check to see if the stock has risen or fallen compared to its old value

QUOTEHISTORY

SYMBOL	QUOTE	TIMESTAMP
IBM	68 1/2	9:27

- Finally, update the QUOTEHISTORY record with the current stock trade

IBM® Software

© Copyright IBM 1995

Additional SQL

IBM

DB2 for
OS/2 & AIX

CASE Expressions

```
    CASE                                           CASE value
        WHEN variable = value1 THEN result             WHEN constant1 then result
        WHEN variable = value2 THEN result             WHEN constant2 then result
        ...                                            ...
        ELSE result                                    ELSE result
    END                                            END
```

Self-Referencing Subqueries

```
 DELETE FROM TABLE1
   WHERE COL2 IN (SELECT COL1 FROM TABLE1)

 INSERT INTO TABLE1 (SELECT * FROM TABLE1)

 UPDATE TABLE1 SET COL1 = NULL
   WHERE COL2 IN (SELECT COL1 FROM TABLE1)
```

AS Clause

```
 SELECT  EMPNO,
                  LASTNAME,
                  SALARY+COMM AS PAY
 FROM EMPLOYEE
 ORDER BY PAY
```

IBM® Software

© Copyright IBM 1995

Additional SQL

IBM

DB2 for
OS/2 & AIX

Call-Level Interface Enhancements

- Extending V1.2 CLI
- New DB2 V2 function support
  LOBs, UDTs, UDFs, Distributed UoW, more..
- Common catalog views
  Function call access to catalog information
- Direct access to DB2 V2 Windows driver
- ODBC V2 support

Read Stability (RS) Isolation

- Added to RR, CS, UR
- Focus on concurrent access to data

and many other syntactic and semantic extensions

IBM® Software

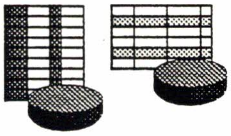
© Copyright IBM 1995

Optimizer

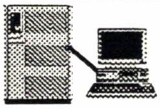
IBM

DB2 for OS/2 & AIX

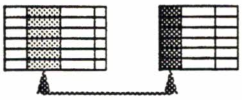
Views



Environment



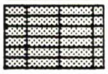
Referential Integrity



Advanced SQL

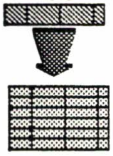
WITH, Recursive SQL
UNION
Sub-selects, JOINs
SQL'92, SQL 3

Constraints



Salary > 20000
Age < 65 and
Service < 40

Triggers



On Update
Set Tax_Code = 3.4 * Average(salary) + Age-12

User-Defined Functions

FLOAT(EXP)
Update_Warehouse(Product, Quantity)

User-Defined Datatypes

Currency, Data_array

Complex Environment!

IBM® Software

© Copyright IBM 1995

Optimizer Technology

IBM

DB2 for OS/2 & AIX

Select Optimization

```
SET CURRENT QUERY OPTIMIZATION = 1
SELECT * FROM EMPLOYEE
WHERE SALARY>50000
```

- 0 = Minimal Optimization
- 1 = Normal Optimization
- 2 = Cartesian Join Optimization
- 3 = Maximum Optimization

Query Rewrite Transformations

- Addition of implied predicates
- General predicate pushdown
- Magic Set transformations
- Existential subquery to join transformation
- Redundant join elimination
- Convert quantified predicates to scalar subqueries
- Convert INTERSECT to EXISTS subqueries
- Conversion of OR to IN
- Conversion of IN to join
- View merging
- DISTINCT elimination

Optimizer Extensions

- Removal of limits
- RID List sorting
- Index ORing
- Increase in execution plan analysis
- Avoidance of Cartesian product
- Improved join size estimation
- Non-uniform distribution statistics
- Fetch I/O statistics
- Different weights for random and sequential I/Os
- Changes to lock intent hints
- Variable CPU and I/O cost estimation

IBM® Software

© Copyright IBM 1995

DB2 for MVS Support

- ✓ ALIAS for TABLE and VIEW names
- *New!* ✓ AS clause in SELECT list
- ✓ CHAR scalar function with DECIMAL operands
- *New!* ✓ CHECK CONSTRAINTS
- ✓ CHECK PENDING and CHECK DATA utilities
- ✓ CURRENT TIMEZONE special register
- *New!* ✓ Nested table expressions
- ✓ DIGITS scalar function
- ✓ Distributed Unit of Work
- ✓ DOUBLE PRECISION synonym for FLOAT
- ✓ DRDA Application Server
- ✓ DRDA bind options
- ✓ Escape clause on LIKE predicate
- ✓ INSERT supports subselect on LONG strings
- ✓ UNION ALL will operate on LONG string
- ✓ User updateable CATALOG statistics
- ✓ Reserved word list reduction
- ✓ IBM sample database
- ✓ INTEGER constant always a large integer
- ✓ INDICATOR keyword
- *New!* ✓ REBIND package command
- ✓ SQLSTATE changes
- ✓ SUBSTR result can be a LONG string
- ✓ EXPLAIN statement
- ✓ DB2 for MVS flagger
- ✓ HEX constants
- ✓ HEX scalar function
- *New!* ✓ Stored procedures

© Copyright: IBM 1995

ISO/ANSI SQL Enhancements

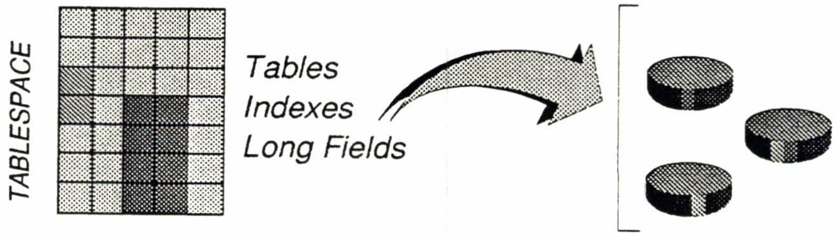
Version 2 Feature	SQL92 Entry	SQL92 Intermediate	SQL92 Full	SQL3
INDICATOR keyword	✓			
AS clause in SELECT list	✓			
SQLSTATE changes to ANSI spec	✓			
Escape clause on LIKE predicate	✓			
DOUBLE PRECISION syn. FLOAT	✓			
Check constraints	✓	✓		
Column naming rules changes	✓			
CREATE/ALTER TABLE changes	✓	✓		
CASE expressions		✓		
AS keyword in FROM clause		✓		
Expression support in COUNT		✓		
Scalar subquery can replace scalar		✓		
WITH CHECK OPTION changes		✓		
Self-referencing INS/DEL/UPDATE			✓	
Extensions to VALUES clause			✓	
Expressions in IN lists			✓	
Set operation result data type rules			✓	
>1 column function with DISTINCT			✓	
Nested table expressions			✓	
User-defined external functions				✓
Set CURRENT FUNCTION PATH				✓
User-defined distinct types				✓
Large objects (BLOBS)				✓
Triggers				✓

© Copyright: IBM 1995

Tablespaces

DB2 for OS/2 & AIX

- Allows tables, indexes and long fields to be separated
- Data can span both physical and logical volumes
- New Backup and Recovery options
  - Allow multiple back-up devices to run in parallel
  - Use of non-standard devices (CD-ROM)
  - Exploit parallel I/O for faster restore/load
  - Use multiple backup buffers
  - Selective Backup and Restore

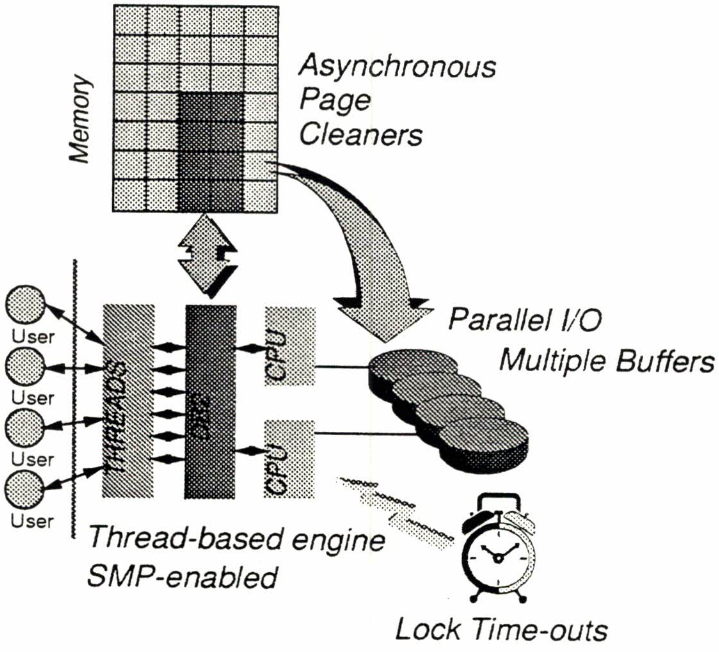

TABLESPACE
Tables
Indexes
Long Fields

IBM® Software

© Copyright IBM 1995

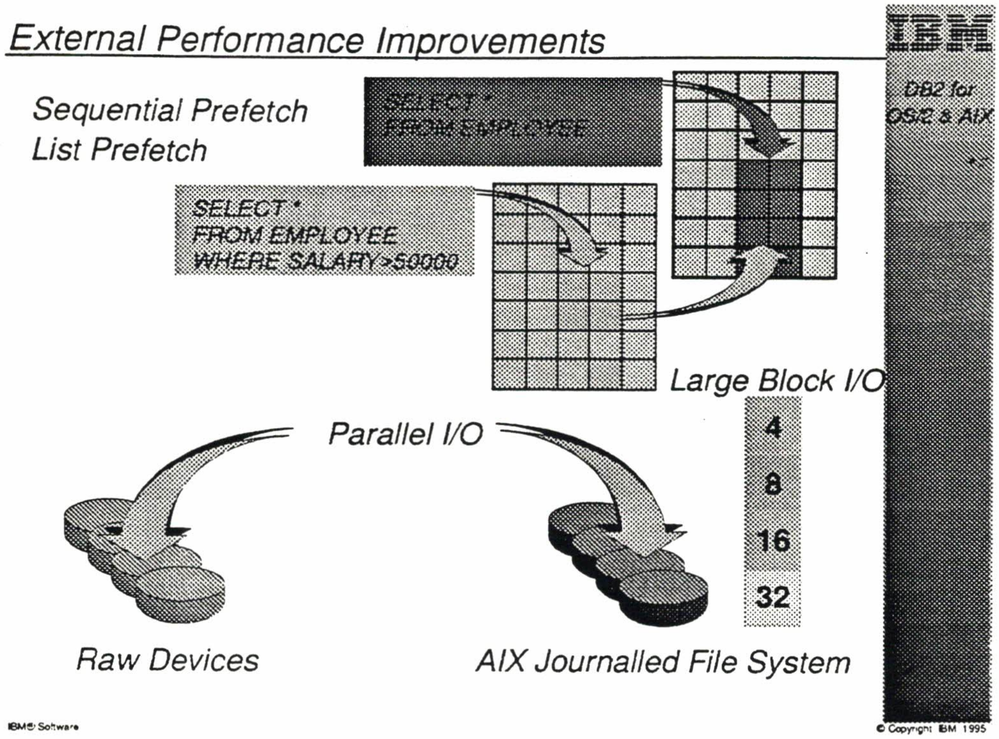
Internal Performance Improvements

DB2 for OS/2 & AIX


Memory
Asynchronous
Page
Cleaners
User
User
User
User
THREADS
DB2
CPU
CPU
Parallel I/O
Multiple Buffers
Thread-based engine
SMP-enabled
Lock Time-outs

IBM® Software

© Copyright IBM 1995



External Performance Improvements

DB2 for OS/2 & AIX

Sequential Prefetch
List Prefetch

SELECT *
FROM EMPLOYEE

SELECT *
FROM EMPLOYEE
WHERE SALARY>50000

Large Block I/O

- 4
- 8
- 16
- 32

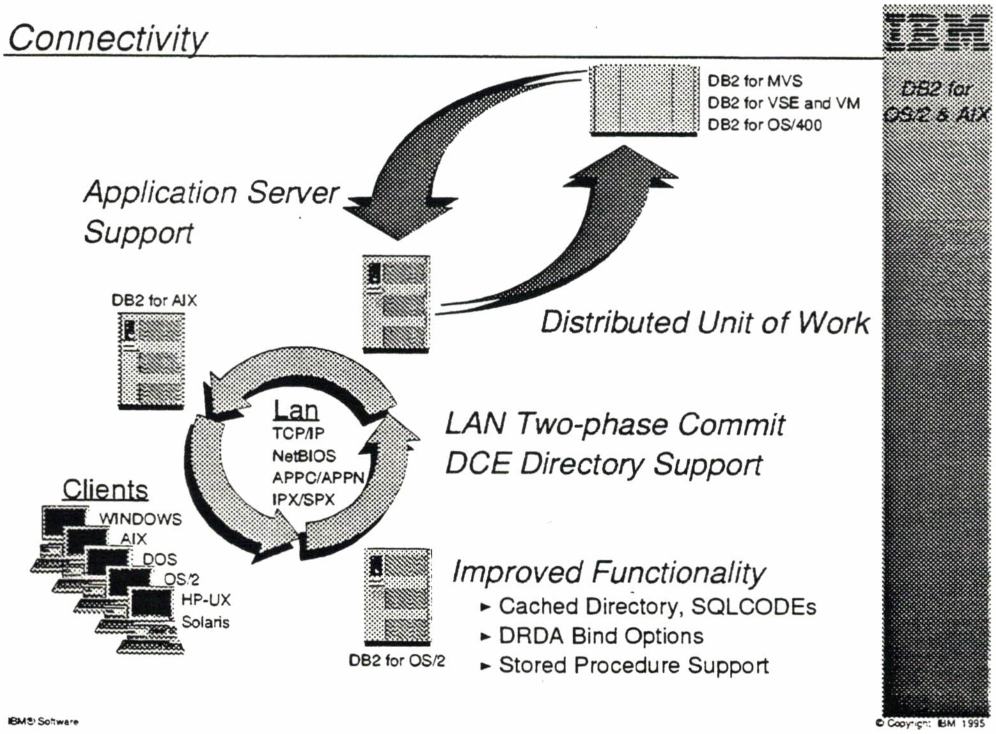
Parallel I/O

Raw Devices

AIX Journalled File System

IBM® Software

© Copyright IBM 1995



Connectivity

DB2 for OS/2 & AIX

DB2 for MVS
DB2 for VSE and VM
DB2 for OS/400

Application Server
Support

Distributed Unit of Work

DB2 for AIX

Lan
TCP/IP
NetBIOS
APPC/APPN
IPX/SPX

LAN Two-phase Commit
DCE Directory Support

Clients
WINDOWS
AIX
DOS
OS/2
HP-UX
Solaris

DB2 for OS/2

Improved Functionality

- Cached Directory, SQLCODEs
- DRDA Bind Options
- Stored Procedure Support

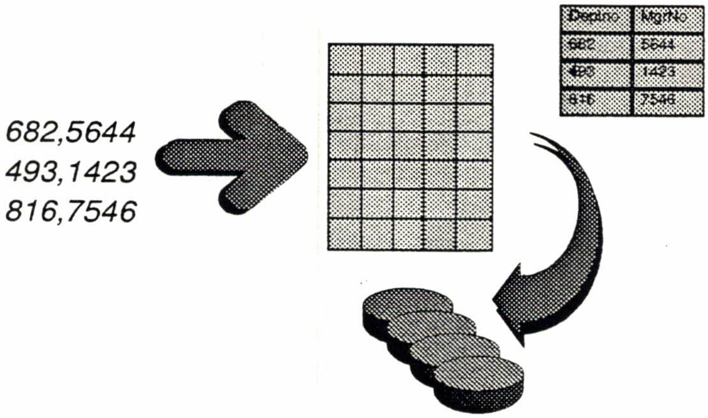
IBM® Software

© Copyright IBM 1995

Load Performance Improvements

DB2 for OS/2 & AIX

- 10 times faster than current IMPORT
- Loads all datatypes, including long objects
- Update statistics
- Create indexes (during load)
- Restartable and recoverable


682,5644
493,1423
816,7546
Deptno
MgrNo
682
5644
493
1423
816
7546

IBM® Software

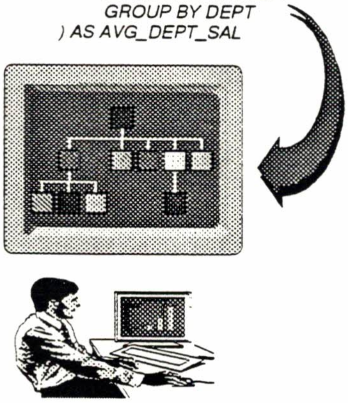
© Copyright IBM 1995

Additional Tools

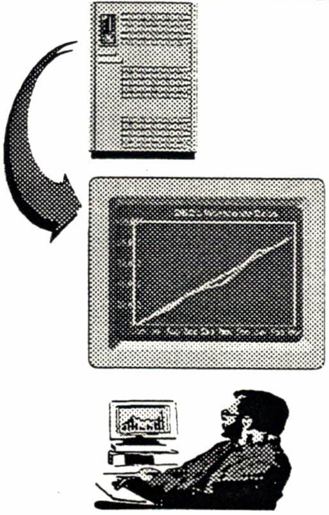
DB2 for OS/2 & AIX

Visual Explain

```
SELECT AVG(AVG_SAL)
     FROM (
          SELECT DEPT,
               AVG(SALARY) AS AVG_SAL
               FROM DEPARTMENT
               GROUP BY DEPT
          ) AS AVG_DEPT_SAL
```



Performance Monitor



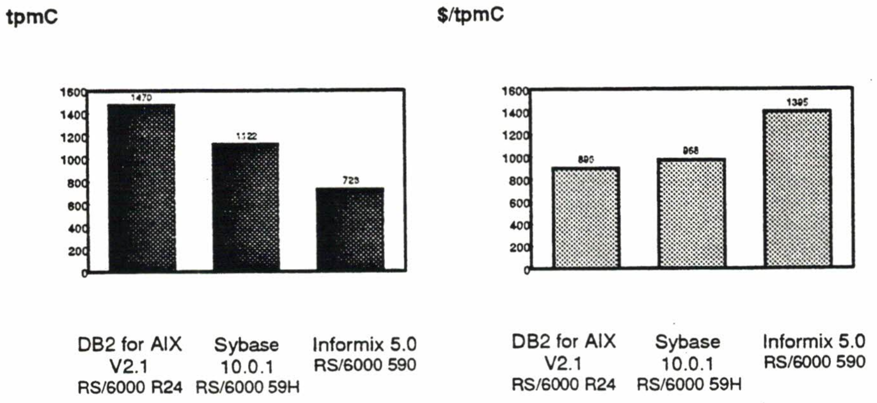
IBM® Software

© Copyright IBM 1995

Leading RS/6000 Performance

IBM

DB2 for OS/2 & AIX



▇ tpmC *

▨ Price/Performance **

On Uniprocessors

* tpmC=Transactions per minute (TPC-C)
** Price/Performance in $/tpmC
TPC-C is a trademark of the Transaction Processing Performance Council

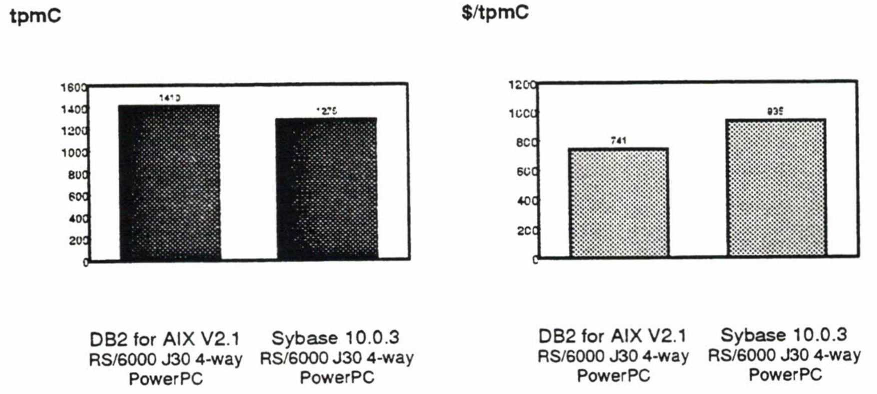
IBM® Software

© Copyright IBM 1995

Leading RS/6000 SMP Performance

IBM

DB2 for OS/2 & AIX



▇ tpmC *

▨ Price/Performance **

On Symmetric Multiprocessors

* tpmC=Transactions per minute (TPC-C)
** Price/Performance in $/tpmC
TPC-C is a trademark of the Transaction Processing Performance Council

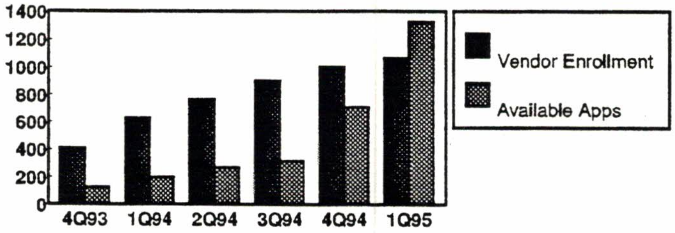
IBM® Software

© Copyright IBM 1995

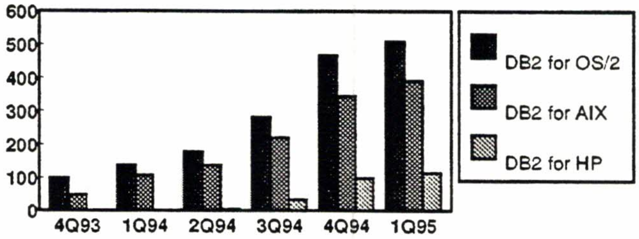
DB2 Developer Assistance Program

DB2 for OS/2 & AIX

Total Applications



Applications by Platform



© Copyright IBM 1995

DB2: A Perfect Fit For AIX and OS/2

DB2 for OS/2 & AIX

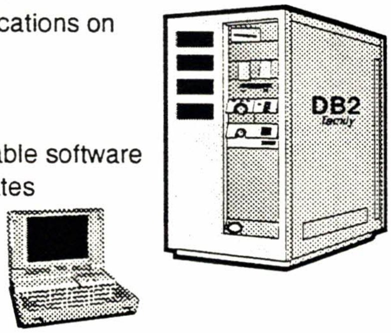
Built on AIX and OS/2
- optimized code to exploit AIX and OS/2 features
- best price/performance for relational database

Superior Connectivity
- excellent connectivity to popular workstations
- support various LAN protocols
- seamless and efficient connection to your legacy data

Open Solution
- mix and match applications and tools
- exploit connection from ODBC applications on Windows
- X/Open standards support

Reliable
- robust hardware combined with reliable software
- built to withstand high transaction rates
- superior systems management

Foundation for Growth



© Copyright IBM 1995

Summary

DB2 for OS/2 & AIX

- ► Version 2 is a significant upgrade to DB2/2 and DB2/6000
  - ▪ Product features aligned between AIX and OS/2
  - ▪ Extended relational functionality
  - ▪ Flexible management of large volumes of data
  - ▪ Improved interoperability through distributed support
  - ▪ High performance OLTP and decision-support
  - ▪ Increased DB2/MVS compatibility
  - ▪ Upward compatibility from Version 1

IBM® Software

© Copyright IBM 1995

DB2 Sources of Information

DB2 for OS/2 & AIX

- ● Compuserve
  - ▪ IBMDB2 Forum
    - ► Section 1 - DB2/2 Questions and Answers
    - ► Section 2 - DB2/6000 Questions and Answers
    - ► Library 1 - DB2/2 Information
    - ► Library 2 - DB2/6000 Information
- ● IBMLINK/TALKLINK
- ● OS2BBS
  - ▪ DB26000 FORUM - Questions and Answers
  - ▪ DB2OS2 FORUM - Questions and Answers
- ● Internet
  - ▪ COMP.DATABASES
    - ► Monitored and questions about DB2/2 and DB2/6000 are answered
- ● 1-800-992-4777 DB2/2 Questions (PMR)
- ● 1-800-237-5511 DB2/6000 Questions (PMR)

IBM® Software

© Copyright IBM 1995

this page intentionally left blank

MGT 0409

OS/2 Warp, Journey Into the Unknown

Ron Cadima
IBM

ABOUT THIS PAPER

Throughout this paper the term Warp refers to IBM's OS/2 Warp Version 3.0 in both the version that operates in conjunction with Windows and the version that contains WIN-OS/2 support, sometimes referred to as the FullPak version.

This paper presents information about Warp that is not generally know or understood by many people. It's purpose is to acquaint the reader with some of the utilities and features of OS/2 Warp as well as how much computer memory is required to support Warp. The memory requirement numbers are based on measurements made on an IBM PS/2 Mod 70-A21 with 16 MB of memory, VGA video support, 120 MB ESDI hard disk, a mouse and 16/4 token ring adapter. HPFS and multimedia support were not installed on the base system. Also, CONFIG.SYS was changed so that DISKCACHE=48 and THREADS=96. The memory measurements will vary if different hardware or software configurations are used, but can be used for general planning purposes.

TRADEMARKS:

IBM Corporation, Armonk, NY

OS/2 Warp Version 3.0
Lan Requester 4.0
Personal Communications / 3270

Stac Storage & Communications, San Diego, Ca.

Stacker Version 4.0 for OS/2 & DOS

Microsoft Corp., Redmond, Wa.

Windows 3.1, Windows 3.11, Windows for Workgroups

MEMORY REQUIREMENTS

The measurements numbers that are presented below are a reflection of what is called working set memory. Working set means how much memory is required to perform a specific function or series of steps. This includes the memory required for both executable code and data used by the operating system, OS/2 Warp, and by any program that is running.

MEMORY REQUIREMENTS:

Function	OS/2 Warp For Windows	OS/2 Warp With WIN-OS/2
Open System Folder	3.6 MB	3.7 MB
Open Command Prompts	3.5	3.6
Start a DOS Session	4.5	4.6
Exit the DOS Session	4.3	4.3
Start a Windows Session	6.3	6.0
Exit the Windows Session	5.3	5.3
Open Productivity Folder	3.5	3.6
Start Enhanced Editor	4.5	4.6

The numbers for DOS and Windows sessions are for full-screen sessions and therefore include the working set memory for switching back to the Warp Desktop to stop the measurement process. Following are numbers for starting a DOS session in a window and the Windows PROGMAN support in a window, also referred to as a seamless session.

Function	OS/2 Warp For Windows	OS/2 Warp With WIN-OS/2
Open DOS Window Session	4.16 MB	4.15 MB
DIR Command in DOS Window	3.00	3.00

Windows Session with Standard Run Mode and Public DDE and Clipboard Support

Start PROGMAN	6.17 MB	5.84 MB
Open Folder Main	3.51	3.50

Windows Session with Enhanced Run Mode and Private DDE and Clipboard Support

Start PROGMAN	5.86 MB	5.69 MB
Open Folder Main	3.39	3.35

The above numbers show that how you set up your session can have a significant impact on the amount of memory used.

The next series of numbers shows how much memory is required when the particular support is is installed or activated on your system. These number are presented as Kilobytes (KB) in size. For example, Save Desktop costs 300KB in working set memory. So, if the desktop was being saved while the System folder was being opened, 3.9 MB of memory would be required instead of 3.6. The numbers below are in addition to those above.

Support	**Additional Working Set Memory**
Multimedia	
Device Drivers	50 KB
Desktop Folder and Objects	100
System Sounds	40 - 300 (Inactive - Active)
Save Desktop	300
HPFS	200 - 400 (No HPFS Partition Defined, and 64 K cache)
XDF Support	50
Personnel Communications 3270	
Support not started	<20
Support active	
Minimum	300 (Single session and does not
Maximum	450 include connection protocol)
TCP/IP	500 (Base support)

This is for the Device Drivers that are installed with the BonusPak.

Lan Requester 4.0	1 MB
Device Driver and Setup	700 KB
Requester Active	300

Default install for Token Ring including LAN Messaging support.

Stacker 4.0	150 KB (Accessing non-stacked partition)

UTILITIES

Let us now look at some utilities that are available in Warp and what they are used for.

RMVIEW

This is the Warp system resources viewing utility. It allows you to view what hardware adapters are installed in your system, what device drivers are defined in CONFIG.SYS to support the hardware adapters, what IRQ (Interrupt Levels) are in use, the DMA channels being used, and which I/O hardware port addresses are being used. This information can be used to resolve any hardware conflicts between 2 or more adapters, to insure that newly installed hardware is configured correctly, and to see which resources are being used. It is especially helpful to insure that installed software, such as communications or multimedia support, has the proper hardware settings defined in it. It can also tell whether 2 different hardware adapters, such as a sound card and a game adapter, are trying to access the same hardware address or interrupt levels.

RMVIEW has a number of different options defined for it. In brief, they are as follows:

/?
Typing RMVIEW /? will display all of the functions that can be used

/P1
Physical view of the planar chipset.

/IRO
This will display all of the interrupt levels that are being used in the system

/DMA
Displays the DMA Channels that are assigned to specific devices.

/IOA
Displays the I/O port address that are assigned to devices.

/MEM
Will show the memory regions that have been claimed by specific devices.

/D
Shows the registered device drivers. These are the newer device drivers which have been written to the Plug and Play standards and register their resource usage when during the initialization process.

/IO
Displays the device drivers that are above 100 HEX.

/HW
Gives a tree view of the hardware installed in a specific machine.

/P
This is the default when parameters are entered and displays all the above mentioned information.

RESERVE.SYS

RESERVE.SYS is used in conjunction with the resource manager in two separate scenarios:

1) If you are using a device driver that is not resource manager aware and you know the resources that are being used by the device driver, you can use RESERVE.SYS to reserve exclusive use of those resources by the non-resource aware device driver.. An example of this type device driver is any DOS or old 16 Bit OS/2 device driver.

2) If you have a piece of hardware that does not tolerate its resources being examined by different device drivers, you can reserve the resources so that the resource manager aware drivers do not examine the hardware.

To use RESERVE.SYS, place this statement as the first line in the CONFIG.SYS file

BASEDEV=RESERVE.SYS <ARGUMENTS>

Arguments:

Argument	format	example	description
/IO:	/IO:x,x	/IO:340,4	Reserve IO ports. First number is base port in HEX, followed by length (number of ports) in decimal.
/P:			Same as /IO:
/MEM:	/MEM:x,x	/MEM:CA00,1000	Reserve Memory. First number is base memory address (HEX), with the assumption that the address is XXXX:0, followed by the length (number of address) in decimal.
/DMA:	/DMA:x	/DMA:2	Reserve DMA Channel. Number is decimal.
/IRQ:	/IRQ:x	/IRQ:13	Reserve IRQ. Number is decimal.
/EXC	/EXC	/EXC	Exclusive resource attribute
/MUL	/MUL	/MUL	Multiplexed resource attribute
/SHA	/SHA	/SHA	Shared resource attribute
/DW:	/DW:x	/DW:10	Decode width of IO address. Valid numbers are 10 and 16. Only valid with /IO:

Note:
More than one resource attribute per resource entry is an error and not allowed. If no attributes or decode width is set, the default is EXCLUSIVE and 16.

Example:
Want to reserve IRQ 13 EXCLUSIVE, DMA 0 SHARED, MEMORY CA00:0 for 1000 bytes shared, IO ports 340 for 10 ports EXCLUSIVE and decode width 16, and IO ports 300 for 64 ports, with a decode width of 10:

```
BASEDEV=RESERVE.SYS /IRQ:13 /DMA:0 /SHA /MEM:CA00,1000 /SHA /IO:340,10
/IO:300,64 /DW:10
```

INSTSND.CMD AND DINSTSND.CMD

When the multimedia support is installed with OS/2, support for generating system sounds is automatically included. System sounds are the noises that are made when folders are opened or closed, the system shut downs, etc. As seen earlier in the memory requirements, this support cost between 40 and 300K in working set memory. The DINSTSND.CDM command file can be used to uninstall the system sounds support and save you the 40 to 300K in working set memory. Removing system sounds support has no effect on your ability to use the multimedia support. All of the device drivers and sound and video support used by multimedia remain unchanged. If, at a later time, you wish to install the system sounds again, use the INSTSND.CMD to do so.

After either of these command files have been run, you will be required to reboot your system for the changes to take effect.

FORMAT

Most everyone is familiar with the format command. With Warp there has been a change made to the format support for what is called quick format. Quick format is a very fast procedure whereby only the directories of the formatted partition are changed. There is no sector by sector formatting done. To choose the quick format option, use the /Q option on the FORMAT command. For example, FORMAT A: /Q, would perform a quick format of a diskette. The default for disk formatting is quick, whereas for diskettes, it is still the long sector by sector format. To do a long format on a hard disk partition, use the /L parameter. The /Q and /L options will work on both FAT and HPFS partitions.

If you perform a format with the /Q option, do not perform a CHKDSK with the /F:3 option specified. This option will recover all of the files on the partition and you will have lost the benefit of the format. Also, do not use the /Q option if you are formatting a drive to remove sensitive data.

VER

The VER command is used to obtain the version number of your installed OS/2 system. This command can be used in conjunction with the BLDLEVEL command to determine what level of software you have installed in your system. BLDLEVL is used for a single executable (EXE) or dynamic link library (DLL) file to specify which build level that file is at. The BLDLEVEL can be used to determine which files have been updated when a fix has been applied to you system. This information should be supplied when to your support organization when a file is having a problem such as a trap. BLDLEVEL will not show anything on non-System files or applications.

SHUTDOWN

SHUTDOWN is a command file that allows you to perform an orderly shutdown of the system. This is equivalent to selecting the shutdown option from the desktop menu list.SHUTDOWN allows you to write your own procedure for stopping you Warp system in an orderly fashion. It allows you to check if any applications are running and to close them down before proceeding with the shut down of the system. This is very useful for unattended systems that can be IPLed over a LAN connection. If you do not close down all of the active sessions before issuing the SHUTDOWN command, a pop up window will appear asking if the session is to be closed, and shutdown will not continue until a response in made to the prompt.

BOOTDISK

BOOTDISK is a utility that you can use to create a set of 3 diskettes for starting OS/2 Warp. The first diskette is a bootable diskette that can be used to start OS/2 from a diskette if you have a problem with the hard drive. The other diskettes contain most of the commonly used and needed utilities such as FORMAT, COPY, CHKDSK, BACKUP, RESTORE, etc. Also contained in these diskettes is a utility called OSDELETE. This utility should be used to remove OS/2 Warp from your system. It will delete all the files associated with OS/2 and cleanup all the references to them.

START AND DETACH

These utilities are not new to OS/2 but worth mention just for clarification. They are used to start a program running in another session. When the other session has started, you gain control back in the session from which you issued the START or DETACH command. The START command allows you to specify a number of options as follows:

/FS - Full screen DOS or OS/2 VIO application.
/PM - Run the program as a PM application.
/WIN - Run the program in an OS/2 Windows session.
/DOS - Run the program in a DOS session.
/MIN - Run the program in a minimized window.
/MAX - Run the program in a maximized window.
/I - Inherit CONFIG.SYS SET statements.

DETACH does not support any options. It basically will start the requested program in a session that can not have access to the keyboard or the screen. Because of this, you can not start any program that would prompt for input from the keyboard because it will never receive it and wait for ever. Any messages that it may display will automatically be discard by the system. Also, programs that are started detached, do not appear in the Task List window. In order to see these programs, you must use the PSTAT utility.

PSTAT

PSTAT displays process status information, such as current processes and threads, system- semaphores, dynamic-link libraries, and shared memory. PSTAT can be used to determine which threads are running in the system, along with their current status andcurrent priorities. It can also be used to help find out why a given thread is blocked and display detached processes. The process ID can be used in a number of other utilities to stop the program from running, change the priority of threads within the process, and in the TRACE utility to trace the processing of a particular process.

PSTAT provides the following options:

/C	Current Processes (Thread ID, Thread State, Thread Priority, Block ID)
/S	Semaphore Information (System Semaphore information including Module Name, Process ID, Session, No. of References, Semaphore name, etc.)
/L	Dynamic Link Libraries (Process and Session IDs and Library List.)

/M Shared Memory Information
/P: xx Process information for given process ID (xx)

It should be noted that when running PSTAT, it is the current active process and it's thread will have a priority boost applied to it. All other process threads will have the priority that they defaulted at or where at when they blocked. The priority for the thread is recalculated when the thread within the process is unblocked.

EAUTIL

EAUTIL is used to detach, attach and modify extended attributes (EAs) to a file. EAs are used to store information about a file. It can contain comment information, archive information, or anything else a program wants to attach. Warp uses EAs to save the name of a desktop or folder object. This utility is especially helpful when files that contain EAs must be distributed over a network or via a media that does not support EAs. If you wanted to save files on a host machine and then distribute them to other OS/2 systems, you could use EAUTIL to detach the extended attributes from the files, distribute your files and then reattach the extended attributes. When you detach the extended attributes, they are saved in a file. You can modify this file with additional extended attribute information before reattaching it to the file.

EAUTIL is started by typing:

EAUTIL FileName EAFileName /Options

Filename is the name of the file that has the extended attributes associated with it.EAFileName is the name of the file where the extended attributes will be saved or from which the new or modified extended attributes will be read.

The options that can be used are:

/S Separate the extended attributes from the file.
/J Join the extended attributes to the file.
/R Replace the extended attributes with a new set of attributes.
/O Overwrites an extended attributes file with new information.
/M Merge information from extended attributes file into existing extended attributes.

DYNAMIC LIBPATH

New with Warp is the ability to dynamically change the LIBPATH associated with a particular process. This can be done from a command line by using a **SET BEGINPATH** or **SET ENDPATH** command or via program control with the **DosSetExtLIBPATH API** call. Another API, **DosQueryExtLIBPATH,** can be used to query any additions that have been made to the LIBPATH. The LIBPATH is used to tell the system where to find DLL files and printer files.

SETBEGINPATH will attach the new directory names to the beginning of the LIBPATH statement that is specified in your CONFIG.SYS file. ENDPATH appends the directories to the end of the CONFIG.SYS LIBPATH statement. This modified LIBPATH is only valid within the context of the process in which it is defined. If another process is started, it will not inherit the modifications made to the LIBPATH.

Possible uses for these capability are in the areas of program development to allow test programs to use test DLL files. It can also be helpful when accessing network drives. Instead of placing the path for the network drive in CONFIG.SYS, you could add it via the SET BEGINPATH or ENDPATH command after determining that the network is up and active. This can save a lot of time because if the network was down, you would have to wait for an error time out period which could be many seconds in length.

ARCHIVE AND RETRIEVAL

Archive and retrieval is another new feature of OS/2 Warp that provides for saving system critical files and allowing these files to be used to recover from a corrupted system file. It also maintains a maintenance system which allows for a selective install process to be done when the whole system is corrupted and unusable. Archive and retrieval provides the following capabilities:

1. Save up to 3 copies of the OS2.INI, SYSTEM.INI, and CONFIG.SYS file, and then select which one you want to use at system boot time. The system will always save the last 3 copies of these files that were used as long as the option is specified in the Archive field of the Warp Desktop Settings. You can also specify that the Archive and Retrieval menu is to appear every time you boot your machine and how long this screen will remain active. It is recommended that you save these files after you have your OS/2 Warp system set up the way you want it, and then turn off the archive and retrieval feature until you make a change that needs to be saved.

2. The archive and retrieval menu screen supports the following options:

I. Use a backup copy of your system files to boot up the system.

II. Go to a command line before booting up the system. This can be used to modify files, delete or add files, or perform any other operations you may want to do. You can not run PM applications from this session or use any devices other than the disk, keyborad and display.

III. Start the system with the video mode reset to VGA mode. This is especially helpful when you change display adapters and need a new device driver installed, or when the wrong device driver was installed. It allows you to bring up your system and make whatever corrections are necessary.

IV. Restart the system from a maintenance desktop. This basically resets your system files back to a position from which selective install can be done if no backups can be used.

V. Restart the system using one of your 3 archived system file sets.

VI. Start your system with a copy of a CONFIG.SYS file that you select.

You can create up to 24 different CONFIG.SYS files. What you need to do is create a command file called ALTF1?.CMD where the ? is replace by a letter from A to Z. In this command file, you will place the commands necessary to copy the new CONFIG.SYS file onto your boot partition. You can do additional procedures within this file that do not require PM support or devices such as communications, audio, etc. The letters C, M, V, and X are reserved and cannot be used in an ALTF1?.CMD name. This command files needs to be located in the \OS2\BOOT subdirectory.

Also in the \OS2\BOOT subdirectory are three files called ALTF1TOP.SCR, ALTF1MID.SCR and ALTF1BOT.CMD. These files contain the text that appears on the Archive and Retrieval menu screen. You can edit these files so they are meaningful for your work environment. Before these files are edited, you must make them write enable. The ATTRIB command can be used to accomplish this function. For further details refer to the OS/2 Master Help Index online documentation.

DESKTOP CUSTOMIZING

There are many ways available for customizing your OS/2 Desktop. You can write a REXX utility that will add objects to your desktop and modify the settings of other objects. You can edit and compile the resource (.RC) file and run them through the resource compiler. You can modify the INSTALL.RC and PHASE2.RC files located in the \OS2\INSTALL directory to have your changes automatically included when you install an OS/2 system. At the end of this paper you will find a sample REXX program that starts a DOS application and modifies the DOS session settings for the program. For detailed information on customizing your desktop, refer to Bruce E. Hogman's article "Launching Objects on the OS/2 Desktop" in issue 3, 1994 of the Personnel Software Magazine. It will provide further references where additional information can be obtained.

TIDBITS

In this next section I will provide little pieces of information that can be used to help improve your systems performance or make things work properly. For more detailed information on OS/2 performance tuning, refer to my OS/2 Warp Performance Tuningwhite paper that is generally available from the OS2TOOLS library.

PRINTING OPTIMIZATION:

Almost all of the printer device drivers have some sort of advanced printer properties associated with them. The most common are:

Printer Fonts - This tells the system that printer has fonts installed in it and that they should be used whenever possible. If these fonts can be used, there will be significant improvements in both spool time and print time because less data will have to be sent to the printer, and less processing will be done by the OS/2 print manager.

Printer Patterns - This refers to special graphic and shading support that is available on some printers. If you have graphics or different shading on text, you can gain improved performance by selecting this option. Instead of the OS/2 system having to generate an image of the page and sending that to the printer, it just send the characters and commands that are required.

Fast System Fonts - When the Adobe Type Manager (ATM) fonts are installed, a set of printer fonts is also installed. These fonts can be down loaded to printers which support this feature to enhance the performance of printing by just sending down the ATM commands necessary to print the job and not a bit map of the page or document.

Large Buffers - This option can be used on systems which have a large amount of physical memory on them. With this option, the printer subsystem will prepare a page worth of data to be sent to the printer instead of a lines worth of data. This helps performance by being able to run the line between the system and the printer at it's maximum speed.

All of these options are basically valid for text mode operations. If you have text that overlays graphic information, the setting of any of these options may make it so the resulting output does show all the text correctly. If this happens, you will have to turn off any of these optimizing features.

AUDIO SUPPORT

As stated earlier, you should remove the support for system sounds or at least turn them off. This is especially true for systems that are memory constrained.

If you are trying to use audio in a WIN-OS/2 session and it is not working, remove the sound drivers via the

Windows Control panel and install the ones found in the OS2\DRIVERS\XXXX directory. The XXXX will be the name of he sound card that is installed in the system.

CD-ROM SUPPORT

You can potentially gain some performance improvements when reading from your CD-ROM drive by increasing the values of the /C and /M parameters on the IFS=CDFS.IFS statement in your CONFIG.SYS file. This assumes that you have enough physical memory in your system.

FILE SYSTEM

If you are using FAT, you should modify the DISKCACHE= statements in the CONFIG.SYS file. If you have more than 4 MB of memory on your system, define the cache threshold value at 16 or 32. Do this by placing a " ,16 " or " ,32 " after the ,LW parameter. Also, you may wish to replace the D parameter with the disk cache you actually need. In most systems that hove more than 4 MB of memory, 256K is sufficient. You should try different values and see which is optimum for your environment.

It is also a good idea to defrag you FAT partitions periodically. Most DOS defrag utilities that do not disturb hidden files can be used without problems.

Finally, you should partition your disk to separate you data files from executable files and to place temporary and work files in a partition by themselves. This will help lessen fragmentation on your disk.

SET COMMANDS

SET MENUSTYLE=SHORT
Add this set command to your CONFIG.SYS file to reduce the options that are in the object menus. It will remove all the functions that can be done with a mouse, such as move, copy, shadow, etc. This will save a little in working set memory and make the menus appear a little faster.

SET OBJECTSNOOZETIME=0
This set command tells the Workplace Shell how long to keep Dynamic Link Libraries (DLLs) around after they are no longer needed. You would set this to a number greater than 0 if you typically had DLLs that were used by applications that were started and stopped in succession. This helps reduce the overhead of opening and closing them very often.

Sample REXX Program to start a DOS program.

```
/* REXX Test Case to start a DOS application */
/* Load system functions library */
call RxFuncAdd 'SysLoadFuncs', 'REXXUTIL', 'SysLoadFuncs'
call SysLoadFuncs

/* Define values for class name, program titles , location and setgup. */
Classname = 'WPProgram'
Title     = 'QBASIC'
Location  = '<WP_NOWHERE>'
Setup =,
        'EXENAME=C:\OS2\MDOS\QBASIC.EXE;'                           ||,
        'PROGTYPE=DOS;'                                             ||,
        'MAXIMIZED=YES;'                                            ||,
        'CCVIEW=NO;'                                                ||,
        'OPEN=DEFAULT;'                                             ||,
        'STARTUPDIR=C:\OS2\MDOS;'                                   ||,
        'OBJECTID=<' || Title || '>;'                               ||,
        ''
rc = SysCreateObject(  Classname,,
                       Title,,
                       Location,,
                       Setup,,
                       'UPDATE')
Exit
```

Two things to note from the above example are the location WP_NOWHERE, and the UPDATE parameter in the SysCreateObject statement. You can replace WP_NOWHERE the name of a folder object if you wanted to add this object to a folder. As it is coded, it is placed in the no where object and disappears when you reipl your system. The UPDATE parameter is required if you will execute this program more than once. If update is not specified, it will ignore the command if the object already exists.

MGT 0411

Electronic Marketing: Profits & Pitfalls

Rick Chapman
Aegis
Stratford, Conn.

ELECTRONIC MARKETING

Electronic marketing comprises both distribution and demonstration vehicles. Traditional electronic systems include bulletin board systems, both commercial and hobbyist, kiosks, and CD-ROM demonstration disks. New to the mix are home shopping clubs, the internet, and CD-ROM distribution.

Electronic distribution and marketing has been tried in the past, particularly in the channel. Several companies have tried to install networks of computers that could ship applications to reseller sites on demand. To date, all of these efforts have failed. Both technology limitations, and the reluctance of buyers to alter traditional buying habits have contributed to these failures.

Despite its rocky history however, interest in electronic marketing and distribution system remains high. The past year or so has seen an expansion of existing programs and the introduction of new ones. Potentially, electronic marketing offers publishers new opportunities to generate incremental revenue. Still, you need to remain cautious when evaluating these programs and qualify MDF requests carefully. Most BBS and electronic systems remain unproven, and their acceptance by buyers has not kept pace with early predictions.

WHY USE ELECTRONIC MARKETING?

Interest in electronic marketing and distribution has been growing for a number of reasons. Chief among these is:

The opportunity to sell into a technically advanced market. Most marketers make the assumption that a buyer who purchases from a commercial bulletin board system or from a CD-ROM is knowledgeable and comfortable with new technology.

The difficulty of penetrating the US channel. Many publishers hope electronic marketing will allow them to create alternate distribution channels and methods that will allow them to reach buyers more quickly. In turn, they hope this will persuade the distribution system to accept their product's more quickly.

The opportunity to learn more about a market. In theory, many of electronic purchasing should allow higher end user capture rates and the ability to precisely discover who is buying a product and why.

The ability to reach large numbers of buyers quickly. For example, once a product has been posted for sale on a system such as Prodigy, information about that product is immediately accessible to approximately 2M potential buyers.

The ability to precisely focus your marketing efforts. For example, if CompuServe hosts several special interest groups (SIGs) devoted to your product, you know where you have an audience who is using and buying your products.

BULLETIN BOARD SYSTEMS AND ELECTRONIC BUYING SERVICES (EBS)

Bulletin board systems are electronic "forums" designed to allow uses to converse, send E-mail to one another, transfer files, provide support, and allow people with special interests to "meet" electronically. They are run on a wide variety of equipment, from micros to mainframe. There are thousands of hobbyist and nonprofit boards in use throughout the country, with dozens appearing and disappearing each month. Some of these systems have become very sophisticated, and a handful are now selling such items as books and training services to their users. It remains to be seen if and how widespread this trend will become.

On the other hand, commercial systems have begun aggressively marketing both software and hardware reselling services for over the last two years. Virtual stores such as CompuServe's Electronic Mall offer both hardware and software from a variety of publishers and vendors. The return on such ventures for software publishers is at present hard to determine. These systems are new. Information on the amount of software purchased through them, breakdowns of purchaser buying habits and volumes, and basic demographic information are hard to come by.

Of course, this does not prevent these services from requesting substantial MDF dollars. Before handling these dollars over, a publisher should ask the buying service the following questions:

What is the size of the service's subscriber base?

What percentage of the subscriber base uses its buying service?

What percentage of users using the buying service buy something?

What is the average size of the purchase?

What is the average frequency of purchase, i.e., three purchases over the last 12 months, two, etc.

What percentage of buying service purchasers purchase software?

What type of software, i.e., entertainment, utilities, business, etc., do they most often purchase?

What are the total annual sales of software purchased through this service?

What is the total membership of special interest groups (SIGs) or forums based on the publisher's products?

Focus Story: You Said What is Confidential?

Company: CompuServe

MARKET OVERVIEW:

Recently CompuServe, the largest and best know electronic service in the country, pitched a business unit of IBM on the virtues of joining the Electronic Mall. This unit sells software and was interested in testing out the Mall as an alternate sales channel. The Mall is CompuServe's version of a virtual store, where buyers can browse through various offerings from different vendors and buy everything from comforters to toasters. Most of the major commercial electronic services (CompuServe, Prodigy, America On-line, etc.) have established electronic buying services, and are aggressively marketing these to software publishers and hardware vendors.

When totaled up, MDF payments were about 18,000 for basic participation, about 70,000 for a more comprehensive marketing effort (i.e., more and bigger ads in CompuServe, more graphics, direct mailings to CompuServe's subscriber base, etc.). In addition, CompuServe would receive 2% on all sales made through the mall.

To justify this expenditure by IBM, CompuServe gave a marketing pitch that covered many key points. They began with demographics. According to CompuServe, their average user is:

Male (91%)

41.5 years median age

College educated (71.5% hold at least a four year degree)

Has an average household income of 93,000

The average CompuServe household is:

Married (74%)

Includes two or members (this seems rather obvious, but CompuServe wants you to know)

Not big on kids (only 36% has children under 18)

These demographics break down to baby boomer yuppies with a lot of money to spend on themselves, a very desirable market for high tech companies. So far, so good. However, as IBM read further through the proposal, problems became apparent.

CompuServe also provided some basic information about the mall member's buying habits. According to CompuServe, the average purchase in the mall was $75. And 40% of CompuServe's approximately 1.4M members visited the mall, and about 40% of them bought something. That translated to a buying audience of about 250k. Not a bad numbers to market to.

As the presentation continued however, problems arose. The first problem was the charts. These were very nice stacked bar charts that described the Electronic Mall's performance from 1990 to 1993. Number or cash orders, accesses (visits) to the mall, connect time in the mall, were among the items being measure. All well and good. The only problem was the charts had no numbers attached to them, either relative or actual. They simply consisted of stacked bars with no data attached to them. (It might have been possible to develop relative data by sitting down with a ruler and measuring the heights of the graphs, but IBM declined to do this.) Certainly they showed growth, but with no tick points on the axis to indicate what was being measured, and to what scale, it was impossible to show what kind of growth was occurring. In practical terms, they were useless.

The second problem was the lack of hard information about what kind or products CompuServe buyers were buying. It was not an issue of CompuServe not knowing. Because of the electronic medium in which a EBS exists, it can provide precise breakdowns of its customer base's buying habits an Egghead or CompUSA can only dream of. CompuServe proved unwilling to provide this information until prodded, and the figures provided were "unofficial."

The final problem occurred when IBM decided to cut to the chase and ask the most important question of all: How much software did buyers purchase through the mall on a yearly basis? IBM did not ask who sold what, simply the aggregate total. CompuServe responded that this information was "confidential." Incredulous, IBM said that no other channel partner would ever respond that way. Resellers and distributors are expected to provide gross information about their hardware and software sales. Most go even further, and provide breakdowns of sales into specific market areas. CompuServe responded that they understood that, but that the information was still confidential. At that point IBM told CompuServe they would take the mall under advisement.

LESSONS

The important lesson of this story is the need to ask detailed questions about new and alternate buying channels. Initial projections and sales pitches are always rosy, but because these service are so new, hard data and sales histories may be lacking. A great deal of initial skepticism is called for.

It is important to also keep in mind that this story does not demonstrate that publishers should not use EBS as part of their marketing and sales mix. A publisher could certainly infer from CompuServe's unwillingness to state software sales that currently not much software is being sold by the Electronic Mall. There could be several reasons for this. One is that people just don't care for buying in that environment. Another is that the concept is new, and will take time to gain acceptance. A third reason could revolve around what type of product is best sold in this type of selling environment.

When faced with these types of unknowns, a publisher must be aggressive in looking out for its 0own self interest. If an opportunity looks interesting, consider doing the following:

Negotiate MDF participation. Ask to tie MDF funding to performance goals. For instance, if an EBS service claims a certain percentage of buyers use their service, ask them to provide reports on usage during your tenure in their service.

Ask for a test period of participation to determine run rates and sell through.

Ask for references. If other companies are using an electronic marketing vehicle, ask for the name of the vice president of sales or marketing of a company selling a noncompetitive product into the same type of audience you've targeted. Call them up and talk to them.

KIOSK SYSTEMS

A kiosk system is electronic display marketing designed to be installed in a retail location. These such units can display text and graphics screen resolution about products and services. The interface uses a menuing system that allows the shopper to view different screens describing various products. These can include:

Product features

Special offers

Reviews and endorsements

Demo screens

Kiosk systems have the potential to complement retail sales efforts in various ways:

As a customer demo and education tool, it can act as a surrogate sales representative.

Sales personnel can use the system to keep abreast of product features and releases

The system can help persuade retail personnel to push particular products.

The system can be used to special-order non-stocked inventory items.

A kiosk can enhance or replace printed literature. It is very difficult to keep retail stores stocked with collateral.

The kiosk can track Inquiry requests and product feedback. This can provide valuable information about market interest in a product, and interest in complementary products.

Despite these advantages, use of kiosk systems in retail stores has been slow. The most well known provider of these services, Sellectek, recently went out of business. The Sellectek system was based on downloading demo and graphics files from a central location to the store-based kiosks. However, other companies, such as SelectWare, are installing CD-ROM-based kiosks in stores. Instead of downloading demos from a remote location, stores receive a CD-ROM that is placed in a demo computer on the store floor.

CD-ROM DEMONSTRATION SERVICES

A growing trend in electronic marketing is CD-ROM demonstration and subscription services. Perhaps the best example of this type of service is Selectware's Software Jukebox. The Jukebox is a CD loaded with product demos and information and sold at retail through resellers. There are currently several versions of the Jukebox, all aimed at different market segments. In addition to product demos, several free applications are included in order to give the Jukebox a retail value. Most of the free applications are older or limited versions of popular games or education packages. Publishers pay Selectware a fee for appearing on a Jukebox. Fees currently range from $1500 to $3000, depending on the amount of data to be placed on the CD and the marketing "collateral" accompanying the demo. Publishers can choose to have their products appear on a quarterly, biannual or annual basis.

HOME SHOPPING CLUBS

One of the newest approaches to electronic selling is offering product through cable TV home shopping programs. While these types of programs have sold computer goods in the past, most of these goods have been overstocks and liquidation items. However, recently several publishers have reported good success with introducing new applications or upgrades through these programs. Typically, the discounts given to these programs match or exceed the discounts a publisher grants to the largest distributors or resellers.

CD-ROM DISTRIBUTION

In the past few months several companies have announced new CD-ROM distribution programs with much fanfare. All of these programs contain the following elements:

A CD-ROM disk containing applications and demos is sold to potential buyers, either through resellers or by subscription.

Most of the applications are locked. The user may try the application or view a demo. If they like it, they can call the CD distributor or publisher to obtain an unlocking code.

The price of the CD-ROM is usually low. New CD-ROMs can be bought singly, or in some cases, as part of a subscription program. Some free applications may be offered on the CD as purchasing incentives.

After a product is purchased, the CD distributor or publisher ship the buyer the documentation and, in some cases, disks.

The Channel and CD-ROM Distribution

Of the major distributors, both Merisel and Ingram have announced CD-ROM initiatives. However, when questioned, these organizations are unwilling to provide details on exactly how these programs will function. The result is that the role of the existing channel structure in CD-ROM distribution is unclear. Industry observers are asking the following questions:

Why should a reseller sell a 49.95 (a typical price) CD-ROM loaded with individual applications with SRPs as high as $995 (the limit for most retail software) instead of the software itself? Will the volume of CD-ROM sales make up the loss of sales of regular applications?

The programs announced by the major distributors compensate resellers for sales of products purchased from CD-ROM. (No details are currently available.) Exactly how do the distributors plan to track sales and compensate resellers?

Are resellers ready to undertake the burden of tracking and fulfilling CD-ROM purchases? Ready to provide buyers with unlocking codes? Willing to ship manuals to buyers? If not, what part in CD-ROM distribution will resellers play?

Software Publishers and CD-ROM Distribution

Publishers are interested in CD-ROM distribution for several reasons. These include:

The possibility of achieving 100% capture rates on products sold via CD. After all, the user must call up a fulfillment house or the publisher to unlock the product.

The ability to manufacture product more cheaply. In theory, CD products are less expensive to manufacture. This is especially true if the user decides CD-based documentation is sufficient for their needs.

The possibility of reaching a wider audience. Many publishers are considering participating in these programs in hopes of reaching a wider audience. It is difficult for new and smaller publishers to penetrate the distribution system and gain shelf space. The "electronic shelf" created by a CD is potentially a more open environment, where a new publisher can compete on more equal terms.

For larger publishers, the ability to create what are sometimes called "super SKUs" offers intriguing possibilities. A super SKU is a bundle of a flagship product and a library of add-on, niche, and complementary products. In theory, a publisher can quickly create special promotions by simply unlocking specific applications for targeted promotions.

CD-based distribution has its detractors. Skeptics make the following points:

Claims of cheaper product costs are misleading. Since most CD buyers have, to date, requested full documentation (and disks, in many cases), claims of COG savings are exaggerated. In fact, CD-ROMs may lead to further SKU proliferation and more administrative overhead for publishers.

Larger publishers with significant channel presence may decide that a level CD-ROM playing field is not an asset.

Publishers using CD-ROM distribution run the risk of alienating the channel if distributors and resellers believe they are loosing sales to CD-ROM distribution.

CD-ROM is viewed by many potential buyers as a medium more suited to home and entertainment products.

As of this writing, little hard data on the realities of electronic marketing exists. Some approaches will succeed, and some will fail. Be skeptical, negotiate, and stay abreast of what's taking place in the marketplace. It's the only way to insure you don't pour money down a marketing black hole.

MNC 0401

Unleashing OS/2: Getting the Top Performance Right Away

No paper available. Please contact speaker

this page intentionally left blank

MNC 0406

TCP/IP for OS/2

Bob Panza
The Systems Expert
Woodbury, Conn.

SYNOPSIS

TCP/IP software for OS/2 will be discussed within the contexts of TCP/IP architecture. Installation, configuration, and operational aspects will be reviewed. Multiple protocol issues will also be summarily addressed. This session is a takeoff on "Inside TCP/IP for OS/2," published in the March 1995 issue of OS/2 MAGAZINE.

AGENDA

1. Protocol convergence issues
2. Importance of TCP/IP
3. TCP/IP architecture
4. Terminology
5. TCP/IP software packages for OS/2
6. Telnet
7. FTP
8. Other functionality
9. IP addressing
10. Internetworks, ho!

TCP/IP FOR OS/2

Among the many LAN protocols running today on large scale computer networks, the Transmission Control Protocol/ Internet Protocol (TCP/IP) suite prevails as the de facto standard. All computer platforms, including PC, Macintosh, Unix workstation, midrange processor, and mainframe, can embrace TCP/IP. A common protocol suite, such as TCP/IP, helps to alleviate the interoperability woes which characterize heterogeneous distributed computing. The purpose of this article is to familiarize the reader with the importance of TCP/IP and to provide a basic understanding of TCP/IP in general. A forthcoming article will delve into installation, configuration, and tuning specifics.

PROTOCOL CONVERGENCE

As network computing evolved, proprietary protocols proliferated. The Novell NetWare world advocated Sequenced Packet Exchange/Internet Packet Exchange (SPX/IPX). IBM ran Data Link Control (DLC) and Network Basic Extended User Interface (NetBEUI). Microsoft brought Named Pipes into the fray. Macintoshes were built around AppleTalk. DEC maintained its own extensive array of network protocols. Meanwhile, the Unix world focused on TCP/IP.

Protocol offerings soon overwhelmed physical topologies, and greatly contributed to the difficulty of getting all network nodes to work together. The holy grail of standardization was sought via the International Organization for Standardization/Open Systems Interconnection (ISO/OSI) reference model.

While the ISO/IEC OSI reference model was embraced in the real world at the physical layer and in part at the data link layer, the so called "middle layers" (network, transport and session) remained the dominion of non-standard, proprietary product offerings. Since much of the ISO/OSI model failed to take hold in the middle layers, protocol proliferation lingered and complex corporate interoperability suffered as a result. Network designers and implementors became memory jugglers and interoperability gymnasts balancing multiple protocol stacks on backends, frontends, and so on. The dynamics of the multiprotocol environment overburdened network management resources. Many corporate network empires were strained as though they were under seize by invading protocol hordes.

On the standards front, Department of Defense (DOD) oriented "open systems" contributed TCP/IP as the ad hoc fallback to the ISO/OSI model for clean interoperability at the middle layers. Many driving forces pushed TCP/IP forward as a de facto standard for non-DOD business environments. The time was right, the need was great, and TCP/IP had an appealing diplomatic posture.

Meanwhile, corporate systems downsizing began to transfer legacy systems from mainframes onto Unix-based platforms. The ongoing trend of downsizing has continually increased the utilization and importance of TCP/IP. Universal acceptance of TCP/IP became easy to attain because the suite had proven itself over the years and was not considered to be proprietary in protocols. As a result, TCP/IP won the protocol popularity contest hands down in the eyes of vendors and end user companies alike.

TCP/IP is also remarkably well behaved. As a resilient "good guy" suite of protocols, it does not like to step on other protocol traffic or aggressively hog bandwidth. Its entrance into the multiprotocol arena makes things better, not worse. In addition, TCP/IP works particularly well on large packet oriented networks, including the largest - the Internet. There is consequently great appeal for corporations to universally deploy TCP/IP as the preferred protocol on their private internets as well as for corporate-wide access to the public Internet.

PC TCP/IP PROTOCOL SUITE ARCHITECTURE DIAGRAM

```
  ISO/OSI                                                   NON-DOD
 REFERENCE                                                 PC TCP/IP
   MODEL                     LAYERED STRUCTURE               MODEL

                -----------------------------------------
                ¦ TELNET FTP TFTP SMTP SNMP DNS RIP LPs  ¦
7.APPLICATION &¦NCS FINGER TALK RouteD X-WINDOW RPC NFS  ¦  APPLICATION
6.PRESENTATION ¦  REXEC KERBEROS SOCKET-INTERFACES ETC   ¦
                -----------------------------------------
                ¦   APPLICATION PROGRAMMING INTERFACES   ¦
                =========================================
5.SESSION &     ¦                    ¦                   ¦
4.TRANSPORT     ¦      TCP           ¦       UDP         ¦  TRANSPORT
                ¦                    ¦                   ¦
                =========================================
                ¦              ¦  ICMP    ¦      ¦       ¦
3. NETWORK      ¦    IP        -----------¦ ARP  ¦ RARP  ¦ INTERNETWORK
                ¦                         ¦      ¦       ¦
                =========================================
                ¦         LINK LEVEL INTERFACE           ¦   NETWORK
2. DATALINK     ¦ - - - - - - - - - - - - - - - - - - - -¦  INTERFACES
                ¦  MEDIUM ACCESS CONTROL (MAC) DRIVERS   ¦
                =========================================
                ¦                                        ¦
1. PHYSICAL     ¦ ETHER-  ¦ TOKEN-   ¦ FDDI    ¦ OTHER   ¦   HARDWARE
                ¦  NET       RING                        ¦
                -----------------------------------------
```

CAPTION - The non-DOD Personal Computer (PC) TCP/IP model conjured above boils the seven layers of the ISO/OSI model down to five layers by combining layer 5 with 4, and merging layers 6 and 7. TCP/IP is not a single protocol by any means; it is actually a collection of many protocols and communication application programs - all engineered to work together. Physical layer topologies represent IEEE standards. The data link layer may deploy IEEE, ODI, or NDIS standards (or some combination of all three or any two via piggybacking). MAC (hardware) drivers interface with specific network adapters. The TCP/IP protocol suite sits above the data link layer. It will co-exist with other non-TCP/IP protocol stacks although none are shown in the diagram. The presence of X.25, PPP, or SLIP would modify the illustrated generic network interfaces.

GLOSSARY - SIDEBAR

TCP/IP terminology is extensive and very technically oriented. Some basic terms are summarily defined below, but keep in mind that they really only scratch the surface. A good TCP/IP textbook or manual will provide much more detail.

Address Resolution Protocol (ARP): The network layer protocol which translates a network layer IP address into a physical layer address (sometimes referred to as a hardware or MAC address).

Domain Name System (DNS): A hierarchial system supporting the translation of high level (symbolic) node names to TCP/IP addresses.

File Transfer Protocol (FTP): This application layer protocol facilitates node-to-node transfer of files and directory structures between a FTP server node and a client logged into the FTP server.

FTP Application Programming Interface (API): An API which provides applications with a client interface for file transfer.

Finger Protocol (FINGER): An application layer protocol that supplies an interface for querying the status of a remote host or a user ID on a remote host. FINGER utilizes TCP as its underlying transport protocol.

Internet: The term once meant the Defense Advanced Research Projects Agency (DARPA) Internet. Today, there is a significant commerical Internet presence. In essence, "Internet" refers to the public virtual network which utilizes the TCP/IP protocol suite for node-to-node communication. The Internet should not be confused with an internet, which generically represents any packet switched network or integrated group of packet switched networks.

Internet Control Message Protocol (ICMP): A network layer protocol designed to handle error conditions and control messages pertaining to network communication problems.

Internet Protocol (IP): The protocol providing packet delivery services at layer three for node-to-node communication via connectionless datagrams. "Connectionless" implies that packet delivery is not guaranteed at that layer.

Interoperability: The ability of different computer system environments and/or applications to communicate with each other across a network in a fashion largely transparent to the computer user. ISO documents often use the term "interworking".

Network Computing System (NCS): A programmer tool kit for distributed application system development. NCS facilitates high level program calls similar to RPC. A programming interface to NCS is provided by TCP/IP.

Point-to-Point Protocol (PPP): The standard serial protocol for communication to the Internet. PPP embraces layers 1, 2, and 3 of the OSI model to provide connection control, error correction, compression, and other features lacked by SLIP.

Port: An endpoint supporting a logical connection between remote nodes running the same application system.

Protocol: Protocols define the format and rules for the transmission and receipt of information in a homogeneous or heterogeneous computer network. While the networked computer systems may be dissimilar in architecture, there must be a common protocol stack or protocol gateway between communicating nodes to facilitate interoperability.

Remote Procedure Call (RPC): A high level programming interface which calls subroutines to be run on a remote host rather than locally.

Reverse Address Resolution Protocol (RARP): The opposite of ARP; it translates a hardware address into an IP address.

RouteD: An application layer function that deploys RIP in order to dynamically create and maintain network routing tables.

Routing: A network layer function whereby a network attached device determines where to send a packet based on the packet's destination address.

Routing Information Protocol (RIP): An application layer protocol utilized to continually maintain routing table entries in routers, gateways, and host computers. Information in the routing table indicates the best way to send packets across the network from one node to another.

Serial Link Interface Protocol (SLIP): A non-standard network layer protocol which allows for point-to-point communication via TCP/IP across a serial line. SLIP lacks the complexity and full features of PPP. It is satisfactory for direct computer-to-computer links but can be inadequate in a large dynamic WAN environment.

Simple Mail Transfer Protocol (SMTP): An application layer protocol used for sending and receiving mail. It also provides specifications for functional third party e-mail system interaction and control.

Simple Network Management Protocol (SNMP): A protocol used by network components, such as bridges, routers, and other intelligent network management devices, to exchange network management information.

Socket Address: A port number combined with an IP address which identifies a particular logical connection in use by an application system.

Socket Interfaces: Interfaces available for custom applications to access and thereby utilize TCP or UDP for communication over a network.

Telnet: The standard layer seven terminal application which supports character, block, and graphics terminals under TCP/IP.

Trivial File Transfer Protocol (TFTP): A simplified file transfer protocol that allows single files to be transferred between network nodes but does not facilitate security or directory control like FTP.

Transmission Control Protocol (TCP): A transport layer protocol supporting connection oriented services for node-to-node communication over the network. Connection oriented services are referred to as "reliable" because packet delivery is "guaranteed".

TCP/IP Protocol Suite: A group of protocols developed by DARPA and originally implemented on ARPAnet. ARPAnet eventually evolved into the Internet.

User Datagram Protocol (UDP): A transport layer protocol providing connectionless communication services across a network via datagrams. Connectionless services are sometimes referred to as "unreliable" because the protocol does not ensure that packets reach their destinations.

X.25 Protocol: An International Telecommunications Union - Telecommunications Standardization Sector (ITU-TSS, formerly the CCITT) standard network protocol which may be utilized to establish a TCP/IP connection directly between two remote computers or between two internet connected remote computers. X.25 is not part of the TCP/IP protocol suite; TCP/IP sits on X.25 only if X.25 is deployed in support of TCP/IP communications. The ISO/OSI model includes X.25.

TCP/IP PACKAGES

Leading TCP/IP vendors have offered TCP/IP on OS/2 for quite a few years. Thus, the software offerings are well established although limited in number. TCP/IP for OS/2 packages in alphabetic order are as follows:

LAN Workplace for OS/2, version 3.0 by Novell

Pathway Access for OS/2 by Wollongong Group

PC/TCP Network Software, OS/2 1.3 by FTP Software

TCP/IP 2.0 for OS/2 by IBM

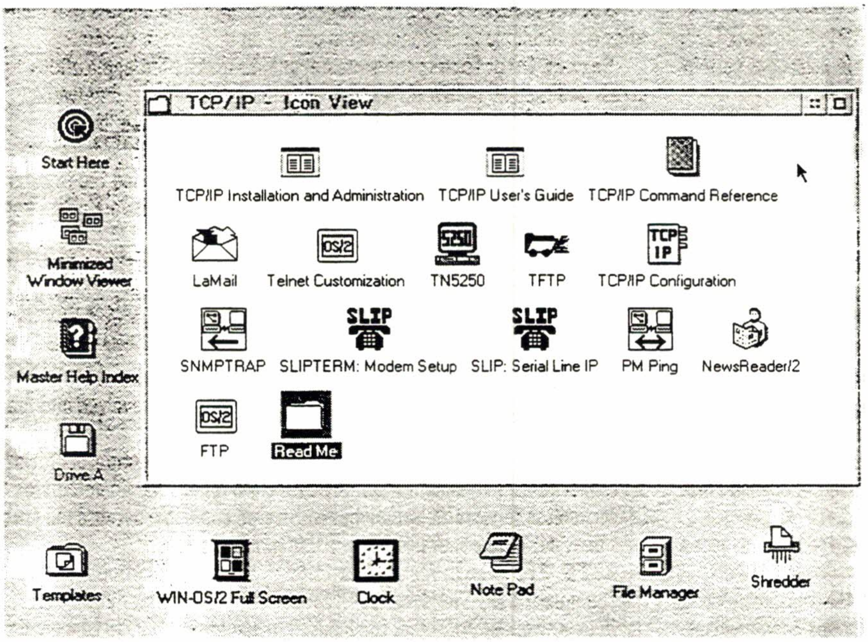
For detailed information on vendor specific TCP/IP product offerings, refer to the LAN MAGAZINE 1994 BUYERS GUIDE ISSUE.

Among the vendors, IBM is the major player since it is the biggest proponent of OS/2. Novell is also an important OS/2 activist because of widespread NetWare connectivity at large end user sites where there is a major IBM and Unix presence. Many OS/2 nodes are backend database or communication servers which run the NetWare Client for OS/2 software. Those backends require support of numerous different protocols in order to deliver enterprise interoperability. Concurrent support across the board must include SPX/IPX, TCP/IP, AppleTalk, and IBM legacy protocols. It is natural for those enterprise accounts to look to IBM or Novell for the TCP/IP piece.

While the underlying TCP/IP protocol suite is fairly standardized, each vendor has put on their own twist of presentation and pizazz. There are also differences in breadth and functionality within each package, although all of the packages maintain the basic TCP/IP features at the OS/2 command prompt. For example, both IBM and Novell offer ASCII Telnet; however, IBM offers Telnet 3270 and 5250 while Novell's LAN WorkPlace for OS/2 does not. In addition, Novell relies on WIN-OS/2 for a graphical user interface (GUI) presentation while IBM uses OS/2 Presentation Manager (PM).

FUNCTIONALITY

A wide range of generic functions are built into OS/2 TCP/IP software packages at the application level, including terminal emulation, file transfer utilities, remote printing utilities, and remote command utilities. In addition, any specific application may run on top of the TCP/IP stack. For example, Lotus Notes and almost all database server-based applications can be configured to run with TCP/IP. The fact that an application sits on and utilizes the TCP/IP stack should be largely transparent to end users.


TCP/IP - Icon View
Start Here
Minimized Window Viewer
Master Help Index
Drive A
TCP/IP Installation and Administration
TCP/IP User's Guide
TCP/IP Command Reference
LaMail
Telnet Customization
TN5250
TFTP
TCP/IP Configuration
SNMPTRAP
SLIPTERM: Modem Setup
SLIP: Serial Line IP
PM Ping
NewsReader/2
FTP
Read Me
Templates
WIN-OS/2 Full Screen
Clock
Note Pad
File Manager
Shredder

The IBM TCP/IP folder displays a wide range of functionality. Traditional TCP/IP features, such as FTP and Telnet, are accompanied by icons for documentation, LaMail (e-mail), Telnet 5250, and so forth. Additional components are available to expand the base offering shown.

Both File Allocation Table (FAT) and High Performance File System (HPFS) are supported for OS/2, DOS, and WIN-OS/2 sessions. (However, one caveat applies: DOS sessions and WIN-OS/2 cannot access HPFS files with long file names.) The OS/2 TCP/IP packages typically present their wares on the desktop in a user friendly manner with pull down menu options, iconized commands, dialog boxes, etc. unique to each package. Mainstay TCP/IP applications, such as terminal emulation and file transfer, can be adapted to work behind each vendor's desktop-based GUI utilities. The look and feel of PM and WIN-OS/2 dramatically transforms the staid TCP/IP command line.

Terminal emulation through Telnet typically supports ASCII terminals, such as TNVT220, VT100, and VT52. The user invokes Telnet by clicking on a Telnet icon (for example, Host Presenter of LAN Workplace) or by executing the command TELNET at the OS/2 prompt. A Telnet prompt will vary depending on the type of terminal emulation invoked (TELNET> VT220> or TN> and so on). After logging into a remote host, the user proceeds to operate the Telnet session as though he or she is using a dumb terminal connected to the remote host. Operations and terminal based applications supported by the remote host may be performed. IBM's TCP/IP software package includes Telnet 3270 (TN3270) and 5250 (TN5250) support to cover the entire IBM terminal world. In addition to the character based TN3270, IBM also offers a PM 3270 emulation package called Presentation Manager Terminal (PMANT). IBM's TN5250 emulation is also a PM application.

Examples of Telnet subcommands are:

?	Lists commands and displays help information. Synonymous with the HELP subcommand.
CLOSE	Closes the current connection with a remote host.
OPEN	Establishes a connection with a remote host and returns to the Telnet prompt.
QUIT	Exits the Telnet command shell.
SEND	Transmits a sequence of special characters to the remote host.
SET	Determines operating parameters.
TOGGLE	Switches operating parameters.

File transfer between TCP/IP networked computers is primarily governed by File Transfer Protocol (FTP) and Trival File Transfer Protocol (TFTP). FTP provides subcommands, security, multiple client support, and handles High Performance File System (HPFS) long file name support. However, TFTP is limited to transferring one file at a time, its daemon services only one client at a time, and it lacks security and subcommands.

"FTP" and "hostname or address" entered at the OS/2 prompt attempt to establish a connection with the host computer running as a FTP server. A FTP prompt looks like the obvious "FTP>". The user must provide identification and password as required by the host. File formats may be converted during transfer. Files can be deleted or renamed on the FTP server. It is also possible for the user to create or remove directories on the FTP host.

Examples of major FTP subcommands are listed below.

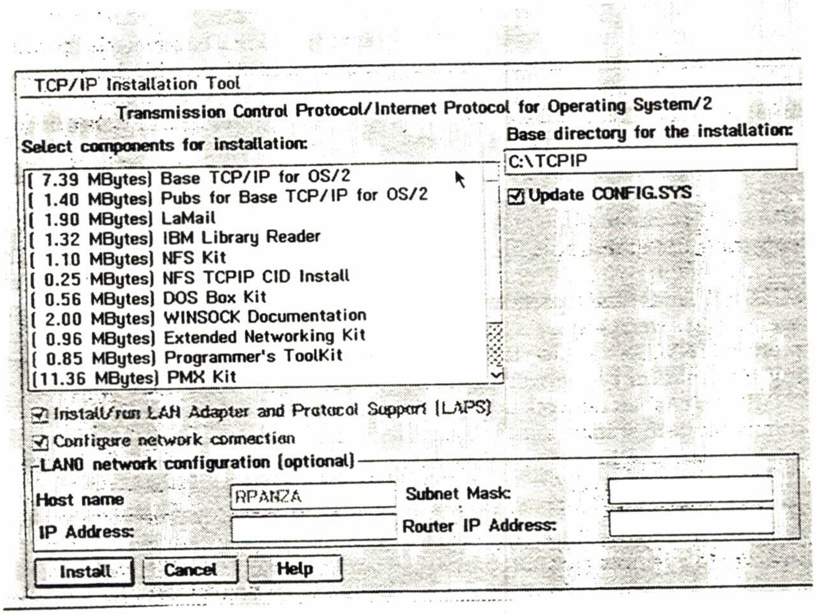
?	Lists commands and displays help information.
ASCII	Sets type of file transfer to ASCII.
BINARY	Sets type of file transfer to binary.
BYE	Ends the FTP session and exits the command shell. Same as QUIT.
CD	Changes working directory on the remote FTP host.
CLOSE	Terminates the FTP session but stays in the FTP command shell.
LCD	Changes or displays the name of the working directory on the user workstation.
LS	Provides a short listing of the working directory's files.
GET	Transfers a specific file from the remote host to the user's workstation.
MGET	Transfers multiple files from the remote host to the user's workstation.
MPUT	Transfers multiple files from the user's workstation to the remote host.
OPEN	Establishes a connection to the remote host.
PROXY	Forwards commands to a server in order to allow a logical connection between another server and it; the objective is direct transfer of files between the servers.
PUT	Transfers a file from the user's workstation to a remote FTP host. Synonymous with SEND.

STATUS Displays status of the FTP connection.

IBM's FTPPM and Novell's File Express are desktop-based GUI applications which make FTPing extremely user friendly.

MORE FUNCTIONALITY

TCP/IP's SMTP protocol supports client and server e-mail. Nevertheless, a SMTP e-mail application must be run on top of SMTP. For example, IBM TCP/IP offers the LaMail user interface for sending, receiving, viewing, creating, and storing mail. LaMail works in conjunction with the Sendmail system. Sendmail actually sits between LaMail and SMTP. Sendmail performs e-mail routing and delivery.



TCP/IP Installation Tool

Transmission Control Protocol/Internet Protocol for Operating System/2

Select components for installation:

```
( 7.39 MBytes) Base TCP/IP for OS/2
( 1.40 MBytes) Pubs for Base TCP/IP for OS/2
( 1.90 MBytes) LaMail
( 1.32 MBytes) IBM Library Reader
( 1.10 MBytes) NFS Kit
( 0.25 MBytes) NFS TCPIP CID Install
( 0.56 MBytes) DOS Box Kit
( 2.00 MBytes) WINSOCK Documentation
( 0.96 MBytes) Extended Networking Kit
( 0.85 MBytes) Programmer's ToolKit
(11.36 MBytes) PMX Kit
```

Base directory for the installation: C:\TCPIP

☑ Update CONFIG.SYS

☑ Install/run LAN Adapter and Protocol Support (LAPS)

☑ Configure network connection

LAN0 network configuration (optional)

Host name: RPANZA

Subnet Mask:

IP Address:

Router IP Address:

TCPINST.EXE, the IBM TCP/IP installation program, reveals optional components: NFS selections, DOS Box Kit, WINSOCK documentation, Extended Networking Kit, Programmers' Toolkit, and PMX Kit. X client selections and Domain Server Kit are out of view below the PMX Kit line. IBM also offers OSF/Motif and NetBIOS-TCP/IP kits.

Line printer commands support remote printing and remote printer management as follows:

LPD	Starts the line printer daemon (LPD) on a network node so that node becomes capable of providing printer spooling services to other network nodes.
LPQ	Allows querying of a remote LPD.
LPRM	Removes a print job running on a remote LPD.
LPR	Transfers a file to a remote LPD.
LPRMON	Redirects the parallel port of a network node to a remote LPD.

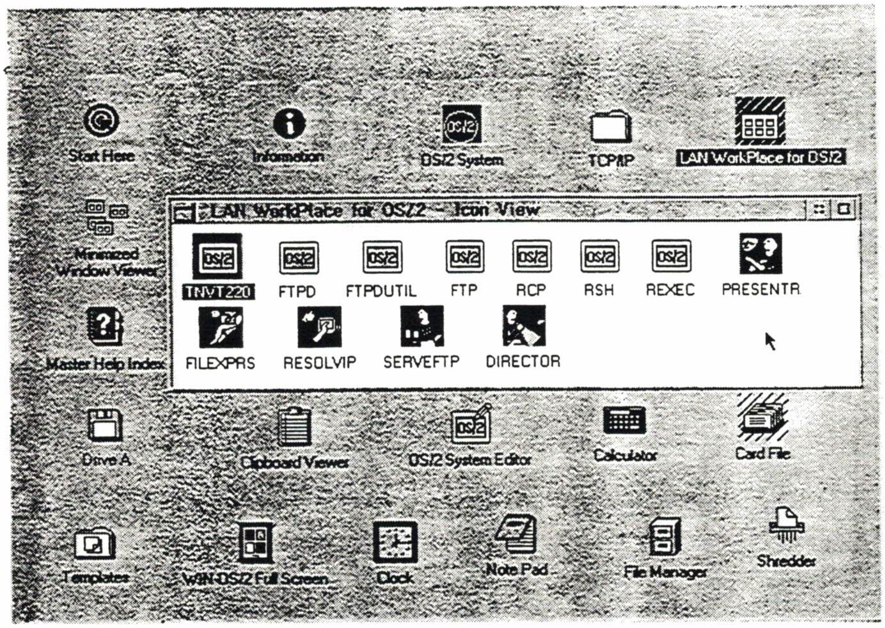
The TALK command utilizes the TCP/IP TALK protocol to send and receive electronic messages. A remote execution (REXEC) command allows a workstation to execute a command on a remote host; password control is facilitated. In addition, the remote shell (RSH) command may be used to execute a command on a remote host; however, RSH does not deploy password control. Remote copy (RCP) is a Unix-like command line program for copying a file or multiple files across the network to or from a remote host.

User level security for TCP/IP communication may be provided on the network by the Kerberos Authentication System. However, a Kerberos server must be running in order to do so.

TCP/IP lends itself to network management in several ways. A specialized protocol, SNMP, is a vehicle for gathering information on network status. SNMP also enables network devices to communicate their status and to enquire into the status of other network devices. The PING command verifies whether or not one computer can reach another computer over the network. PING sends a packet to a remote computer; when the remote computer receives the packet, it returns an Page 11

"echo" packet back to the originator in order to confirm communication. NETSTAT displays status information about the local computer node's network interfaces, packet statistics (for IP, ICMP, TCP, and UDP), routing, connections, and socket port numbers.

The Network File System (NFS) is a common file system which eliminates the need to deal with file system differences across various types of networked computers. NFS utilizes Remote Procedure Calls (RPCs) via UDP for client-server communication. Remote NFS drives may be "mounted" by client computers.

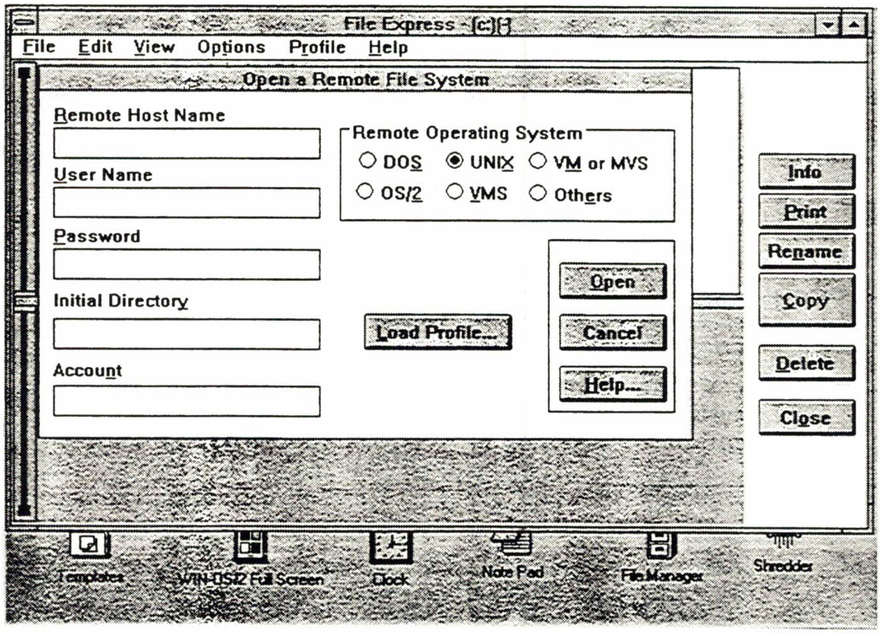

Start Here
Information
OS/2 System
TCP/IP
LAN WorkPlace for OS/2
LAN WorkPlace for OS/2 - Icon View
Minimized Window Viewer
TNVT220
FTPD
FTPDUTIL
FTP
RCP
RSH
REXEC
PRESENTR
FILEXPRS
RESOLVIP
SERVEFTP
DIRECTOR
Master Help Index
Drive A
Clipboard Viewer
OS/2 System Editor
Calculator
Card File
Templates
WIN-OS/2 Full Screen
Clock
Note Pad
File Manager
Shredder

Novell's LAN WorkPlace for OS/2 has user friendly features in addition to traditional TCP/IP functions. PRESENTR provides the user with a dialog box for Telneting into a remote host. FILEXPRS facilitates file transfer through a desktop-based GUI rather than FTP prompt. The local node can quickly become a FTP server via SERVEFTP so remote nodes may access its disk drive.

IBM offers an X Window System. The X Window System is a windows based GUI developed at the Massachusetts Institute of Technology. X window clients connect to X window servers; within the "connection X window", the client can then display and control an application running on the server. OS/2 PM windowed applications and X window client applications may share the same screen because OS/2 PM serves as the X window manager for the server.

IP ADDRESSING

Internet addresses are needed for each network interface. The addresses may be arbitrary if the network will not be part of the Internet. If a network node (or an entire network) will join the commercial (non-Defense Department Network) Internet community, for example, then registered Internet addressing must be obtained from the Internet Network Information Systems Center in Herndon, Virginia. The registration process may be initiated by calling (703) 742-4777 or e-mailing a request to hostmaster@internic.net. Registration helps to prevent duplication of addresses and also helps to maintain an orderly addressing scheme as well as an orderly network.



Remote Operating System
○ DOS ● UNIX ○ VM or MVS
○ OS/2 ○ VMS ○ Others

File Express may be used to access a file system on remote nodes running DOS, OS/2, Unix, VMS, VM, MVS, or another operating system. Interoperability across operating systems seems as easy as clicking a button. FTP security is fully deployed.

The standard internet address format is represented by twelve decimal numeric characters separated by dots, such as 111.222.333.444. (An address appearing as 1.2.3.4 means 001.002.003.004; the leading zeros are not shown.) There are eight bits (one byte) on each side of the dots for a total of 32 bits (four bytes) in the address. Beneath their decimal representation, standard internet addresses actually consist of a two-part address field. The first part contains the network address. The second part holds the local address, also known as the host address, which identifies the specific network node. Within the network address, the high order bit or bits indicate the class of address. Likewise, the most significant bits in the local address may be allocated for subnetwork identification if subnetworks are implemented.

Network node address, router address, and subnetwork mask are vital parts of the network node configuration. The network node's IP address must be unique for specific identification on the network. If a node has more than one Page 12 network adapter, then multiple IP addresses are needed (one for each interface). A duplicate network address is a "no-no" because it raises havoc with packet routing. The local node's default router address identifies the router through which it will send and receive packets when communicating with nodes on remote networks. A remote network is any network on the other side of a router. If subnetworks are implemented, a subnetwork mask identifies how the subnetwork is defined within the IP address.

There are five classes or types of IP addresses: class A, class B, class C, class D, and class E. Class A addresses approximate the form "nnn.lll.lll.lll". The high order bit for a class A address is always set to zero (thereby using one bit) to identify the address as class A; nnn represents the high order bit plus the 7 bit network address. The local node portion of the address, lll.lll.lll, uses the remaining 24 bits. The first byte of a class A address will always be less than 128, for example, 127.lll.lll.lll.

Class B addresses may be depicted by nnn.nnn.lll.lll. High order bits are always set to 10 (2 bits) to flag the address as class B. The high order bits plus the 14 bit network number compose nnn.nnn. A 16 bit local address, lll.lll, accounts for the rest of the 32 bits. The first byte of a class B address will be between 128 and 191.

Class C addresses are represented by nnn.nnn.nnn.lll with the three highest order bits set to 110 (three bits) in order to indicate class C. The high order bits are followed by a 21 bit network number to make up nnn.nnn.nnn, and an eight bit local address, symbolized by lll. Class C addresses range from 192 to 223 in their first byte. IP addresses above the class C range are reserved numbers.

Class A addresses allow comparatively many local nodes within relatively few network addresses since most of the bits are available to define local addresses. Class C addresses define many networks and a relatively fewer number of local nodes within a network. Class B addresses, the most common type of IP address, are on middle ground nearly balancing the allocation of bits between networks and number of network nodes per network. Class D addresses are multicast addresses used for broadcasting to a group of nodes. They are identified by 111 in their high order bits. Class E addresses are beyond the scope of this material.

Administration of IP addresses becomes a nightmare if the situation gets out of control. There is no alternative such as using a burned-in hardware address instead of the IP address. Order or chaos rests in the hands of the system administrator and his or her clan of users.

INTERNETWORKS HO! (CONCLUSION)

As networks grow increasingly pervasive, expanding in breath to potentially link all computing devices, the need for stability in the middle layers becomes an absolute necessity. TCP/IP promises a greater degree of network order and interoperability by streamlining middle layer complexity. Less problematic network environments are the inevitable and highly desirable result. Valuable technical resources can then be redirected to areas which contribute more directly toward the commercial enterprise's success.

PRO 0407

Introduction to the SOMobjects Toolkit

Ira Forman
IBM

Object Oriented Programming

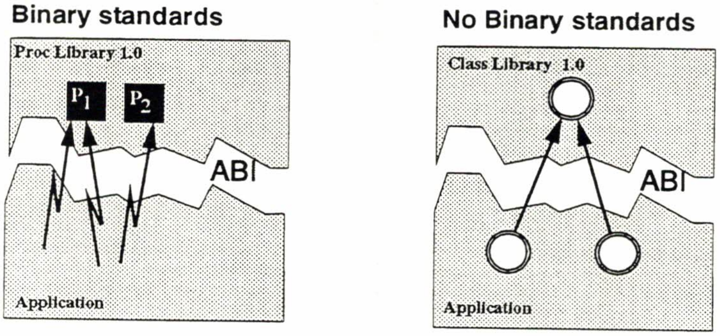
The three pillars of O-O Programming:
- Encapsulation
- Inheritance
- Polymorphism

The Promise:
- Software Reuse
- Increased Programmer Productivity
- Ease of Software Maintenance/Extension

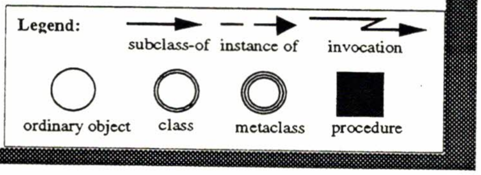
The Reality: DISAPPOINTING
- Languages have been over emphasized
- Packaging of libraries has been ignored
- Single language/single compiler solutions
- Evolution ignored
- Standards non-existent

Why the Difference ?

Subclassing across the Applications Binary Interface (ABI)!


Binary standards
Proc Library 1.0
P_1
P_2
ABI
Application
No Binary standards
Class Library 1.0
ABI
Application

One of the many problems SOM solves!


Legend:
subclass-of
instance of
invocation
ordinary object
class
metaclass
procedure

SOM: Providing the Missing Elements

SOM is a
packaging technology for
object-oriented class libraries.

SOM design goals:

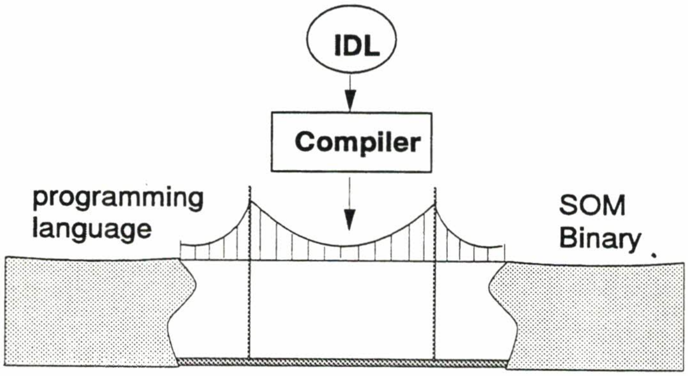
Language Neutral
State-of-the-Art Object-Oriented Capability
Support for Industrial Strength Class Libraries
plus
Support Standards

Language Neutrality

OMG - Object Management Group

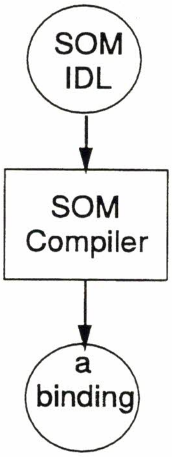
COBRA - Common Object Request Broker Architecture

IDL - Interface Definition Language



SOM Compiler

SOM IDL = **COBRA IDL**
+ Encapsulated Implementation Information

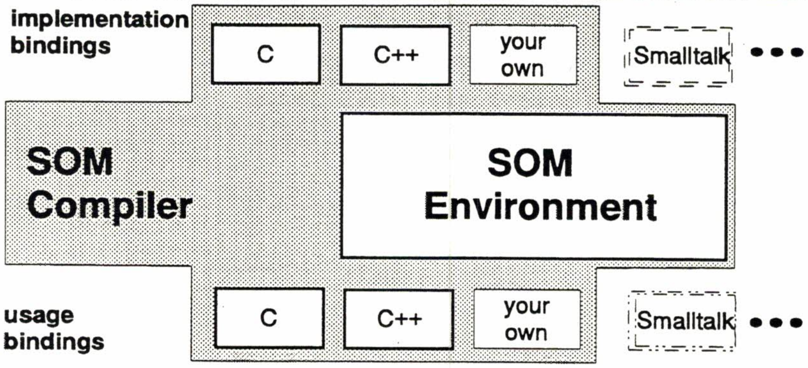

SOM IDL
SOM Compiler
a binding

```
interface dog : <parents>  {
   <interface definition>
   #ifdef __SOMIDL__
   implementation{
      <implementation information>
   };
   #endif
};
```

sc -sxh *dog*

C++ usage binding for *dog*
(in this case a C++ header file)

SOMobjects Toolkit 2.1


implementation bindings
C
C++
your own
Smalltalk
• • •
SOM Compiler
SOM Environment
usage bindings
C
C++
your own
Smalltalk
• • •

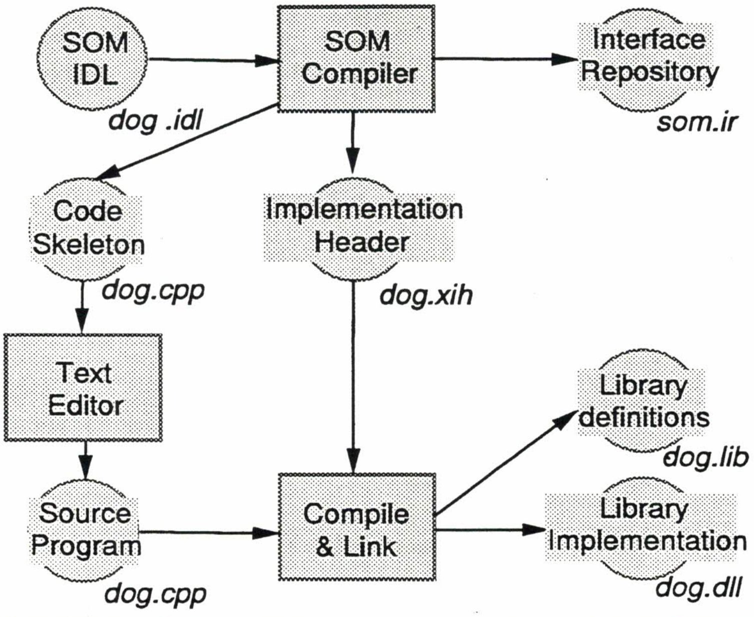
Platforms

• Today	OS/2 Windows3.1	AIX System7	
• Announced:	OS/400	MVS	
• Under consideration:	Windows'95	HP/UX	Solaris

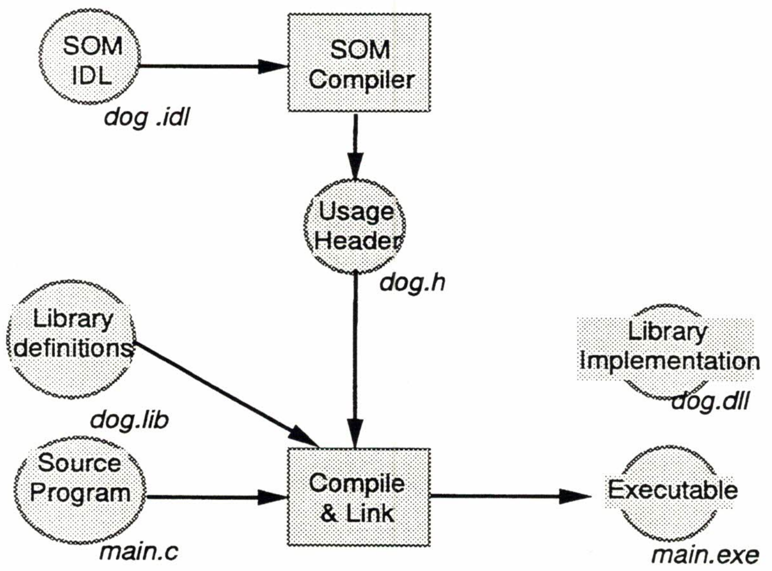
SOM Development Environments

- C/C++ Compilers/IDEs -- Direct-to-SOM
  IBM C Set++
  Metaware High C++
- C/C++ Compilers/IDEs -- Via SOMobjects Bindings
  Watcom C++
  Borland C++
  Symantec
  Microsoft visual C++
- 4GL s & End User Tools
  Watcom VX-REXX
  AM- Intelligent Environments
  Synaptec Object Factory
  IBM OREXX (Direct-to-SOM)
  ORBitize - Netlinks Tech.
  Hockware, VisPro/REXX
- Smalltalk -- Direct-to-SOM
  IBM VisualAge
  Digitalk Smalltalk/V & Parts
  ParcPlace Visualworks
- 3GL Procedural Languages
  ANSI C Compliers
  Microfocus/IBM COBOL
- Announced Tool Support
  Microfocus COBOL
  Object Design
  Inference
  Template Software
  Intermedia
  Kaseworks
  Interactive Objects
  ● ● ●

SOM Class Library Development Process

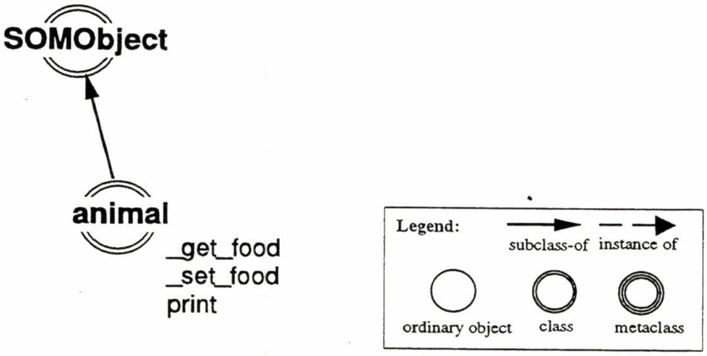

SOM IDL
dog .idl
SOM Compiler
Interface Repository
som.ir
Code Skeleton
dog.cpp
Implementation Header
dog.xih
Text Editor
Library definitions
dog.lib
Source Program
dog.cpp
Compile & Link
Library Implementation
dog.dll

SOM Application Development Process


SOM IDL
dog .idl
SOM Compiler
Usage Header
dog.h
Library definitions
dog.lib
Library Implementation
dog.dll
Source Program
main.c
Compile & Link
Executable
main.exe

A Programming Example

```
#include <somobj.idl>
interface animal : SOMObject {
    attribute string food;
    void print();
    #ifdef __SOMIDL__
       implementation {
          releaseorder: _get_food, _set_food, print;
       };
    #endif
};
```


SOMObject
animal
_get_food
_set_food
print
Legend:
subclass-of
instance of
ordinary object
class
metaclass

SOM compiler creates a method template file...

animal.C file

```
#define animal_Class_Source
#include <animal.xih>

SOM_Scope void SOMLINK print( animal* somSelf,
                              Environment *ev )
{
   animalData *somThis = animalGetData(somSelf);
   animalMethodDebug( "animal", "print" );
}
```

... which you fill in:

edited animal.C file:

```
#define animal_Class_Source
#include <animal.xih>
SOM_Scope void SOMLINK print(animal* somSelf,  Environment *ev)
{
   animalData *somThis = animalGetData(somSelf);
   animalMethodDebug("animal","print");

   somPrintf("\nanimal food: %s\n",somSelf->_get_food(ev));
}
```

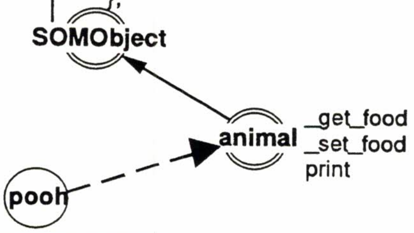
Using the animal class:

C++ Program

```
#include "animal.xh"
main()
{
   animal* pooh;
   Environment *ev;
   ev = SOM_CreateLocalEnvironment();

   pooh = new animal;
   pooh ->_set_food(ev,"Honey");
   pooh->print(ev);

   SOM_DestroyLocalEnvironment(ev);
   delete pooh;
}
```

C Program

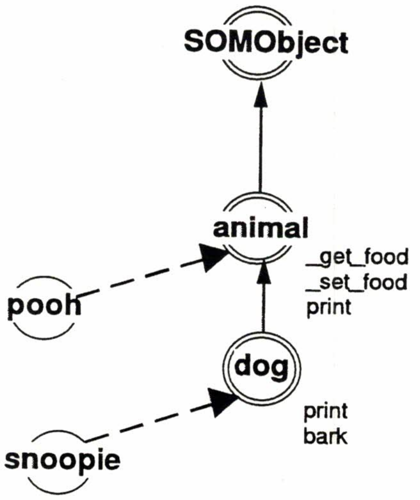
```
#include "animal.h"
main()
{
   animal* pooh;
   Environment *ev;
   ev = SOM_CreateLocalEnvironment();

   pooh = animalNew()
   __set_food(pooh,ev,"Honey");
   _print(pooh,ev);

   SOM_DestroyLocalEnvironment(ev);
   _somFree(pooh);
};
```


SOMObject
animal
_get_food
_set_food
print
pooh

program output:

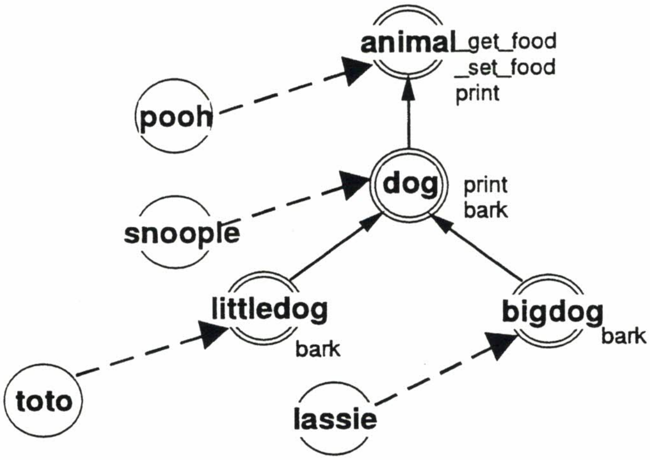
```
animal food: Honey
```

Inheritance (Class Derivation)


SOMObject
animal
_get_food
_set_food
print
pooh
dog
print
bark
snoopie

Multiple Inheritance too!

Polymorphism


animal
_get_food
_set_food
print
pooh
dog
print
bark
snoopie
littledog
bark
bigdog
bark
toto
lassie

animal defines a food attribute and print() method
dog derived from animal, redefines print(), and defines bark()
littleDog is derived from dog, redefines bark()
bigDog is derived from dog, redefines bark()

SOM Design Goals:

- Language Neutral
- State-of-the-Art Object-Oriented Capability
- Support for Industrial Strength Class

Class libraries built with SOM can be
- defined and implemented in one language,
- used from another language.

The class user can...
- instantiate objects
- derive new classes
- override existing methods

Even from a non-object-oriented language!

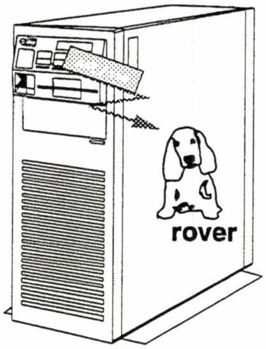
SOM Design Goals:

- Language Neutral
- State-of-the-Art Object-Oriented Capability
- Support for Industrial Strength Class Libraries

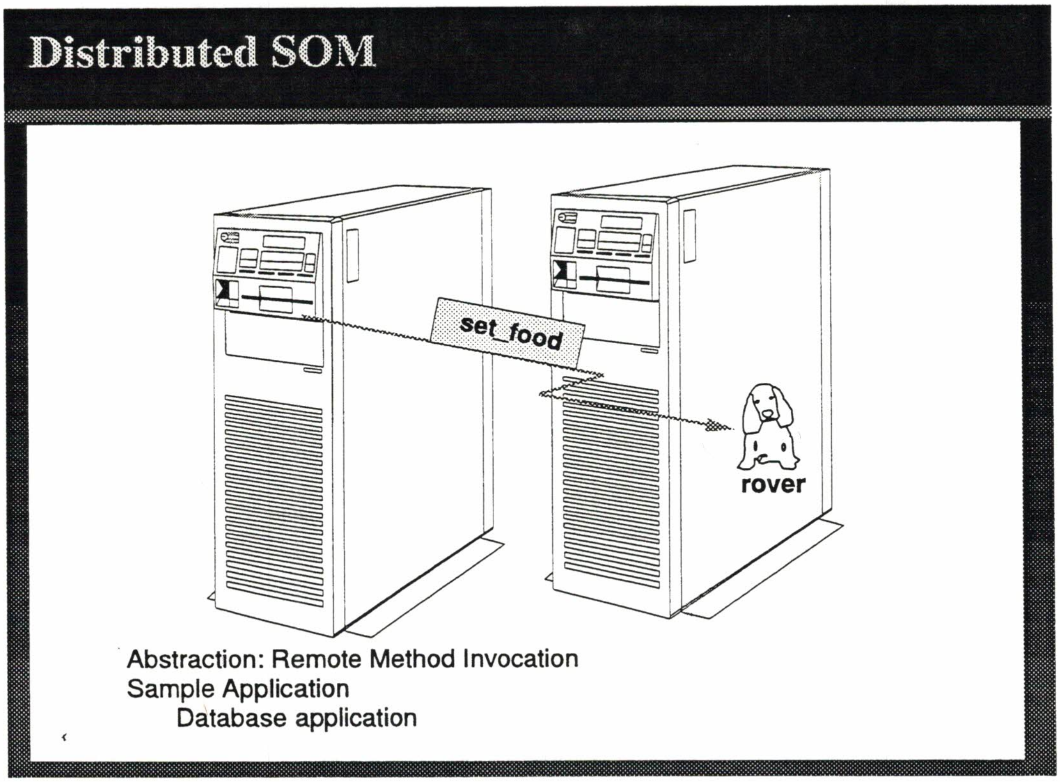
SOMobjects Toolkit:

- Container Classes
- Interface Repository
- Distributed SOM
- Persistent SOM
- Replication Framework
- Emitter Framework
- **Metaclass Programming**

SOM Kernel



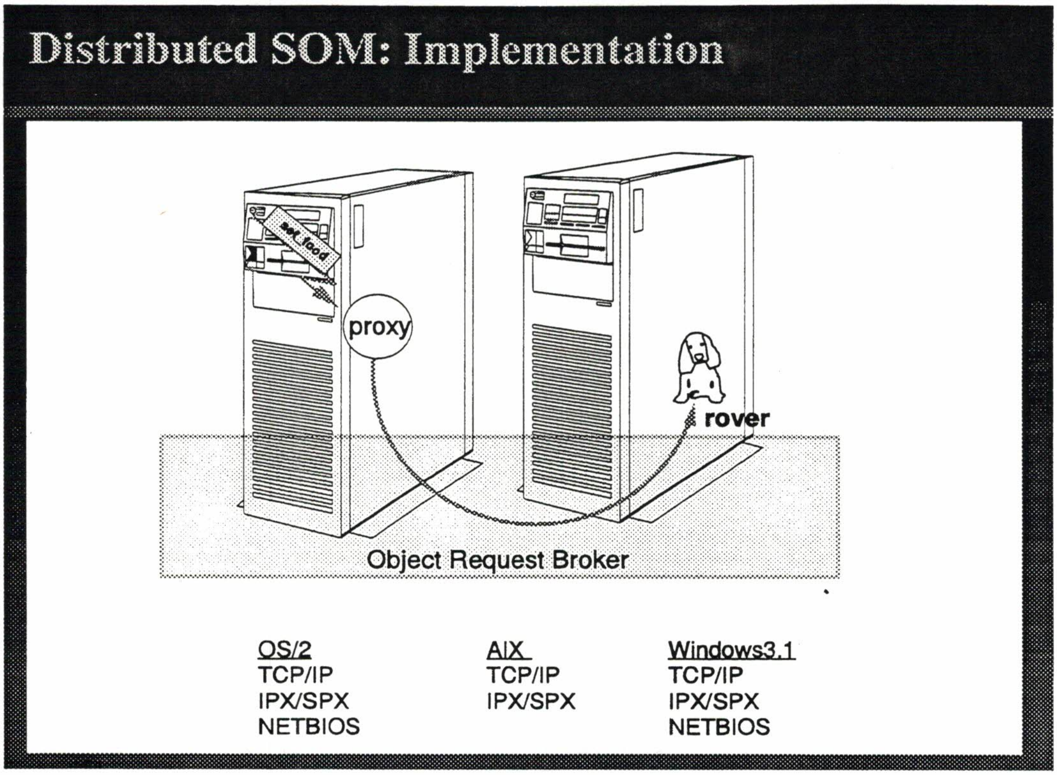
C and C++ Language Bindings
Inheritance, Polymorphism, and Encapsulation
OMG COBRA IDL + encapsulted implementation information
Multiple Inheritance
Sample Application: Developing Class Libraries



Distributed SOM

set_food
rover

Abstraction: Remote Method Invocation
Sample Application
    Database application



Distributed SOM: Implementation

set_food
proxy
rover
Object Request Broker

OS/2	AIX	Windows3.1
TCP/IP	TCP/IP	TCP/IP
IPX/SPX	IPX/SPX	IPX/SPX
NETBIOS		NETBIOS

Distributed SOM

Many SOM class libraries can be distributed without modification

Constraints:

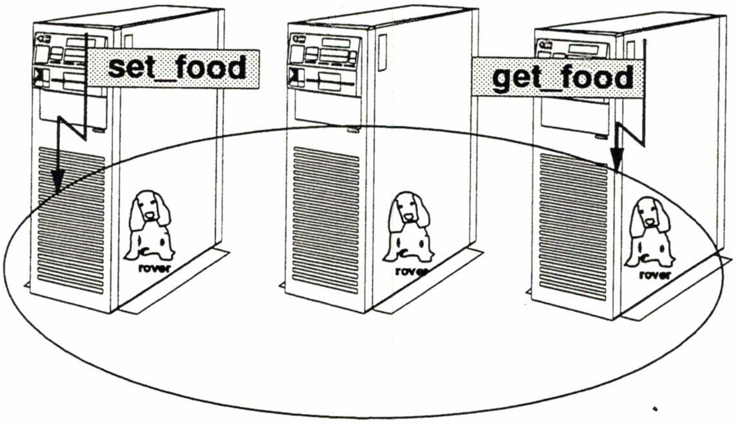
- Must have an IDL interface
- Avoid printing to STDOUT
- Avoid referencing client-allocated memory in instance variables
- Avoid using structs with embedded pointers as method arguments
- If structs or unions are packed or optimized, use IDL modifiers to indicate exact alignment

First 3 constraints: inherent in distributed programming

Second 2 constraints: current DSOM limitation

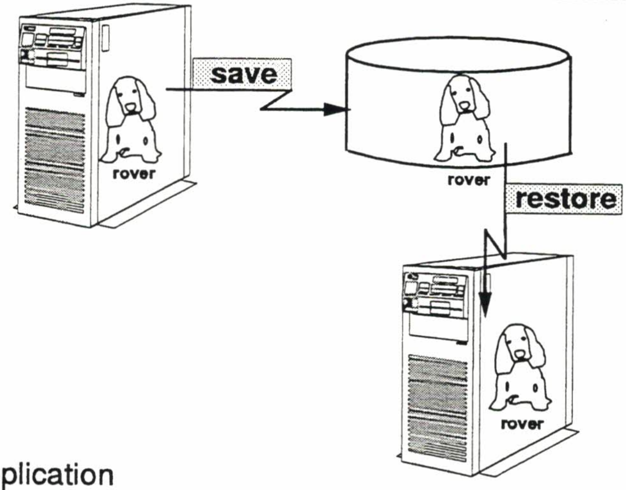
Non-SOM implementations can also be "wrapped" as SOM classes for distribution with DSOM.

Replicated SOM



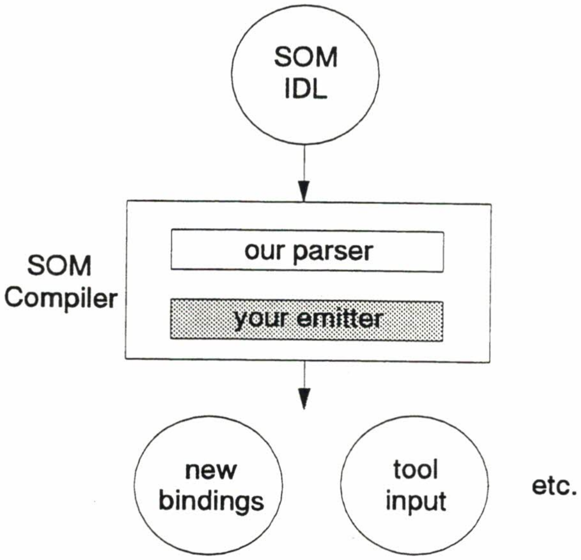
Sample Application: Groupware

Persistent SOM


save
rover
rover
restore
rover

Sample Application
Any in which object state needs to be kept

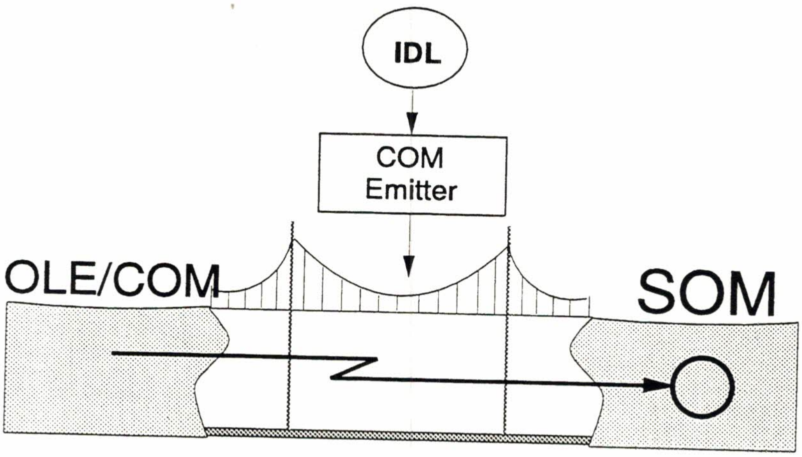
Emitter Framework


SOM IDL

SOM Compiler
our parser
your emitter

new bindings
tool input
etc.

COM Usage Binding Emitter

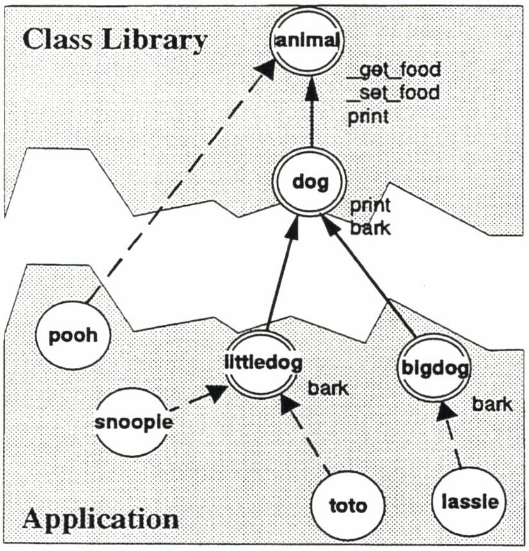


SOM Design Goals:

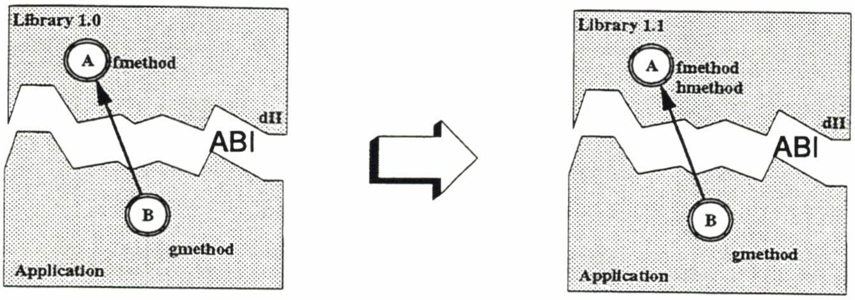
- Language Neutral
- State-of-the-Art Object-Oriented Capability
- **Support for Industrial Strength Class Libraries**

Characteristics of industrial grade class libraries:
- Ship without source code
- Upward binary compatiblilty
- Support shared libraries

What can I do without recompiling?


Class Library
animal
_get_food
_set_food
print
dog
print
bark
pooh
littledog
bark
bigdog
bark
snoopie
toto
lassie
Application

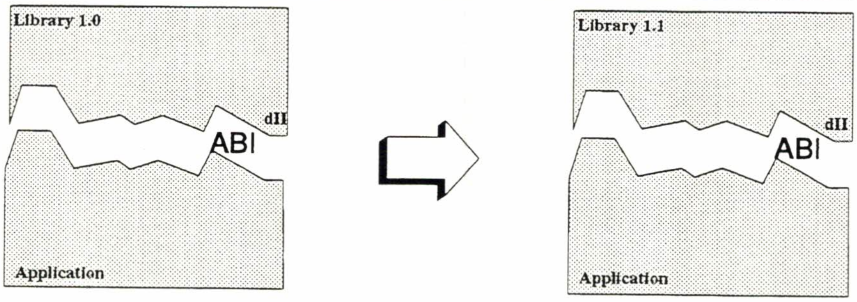
Example for Typical C++ Compiler


Library 1.0
A
fmethod
dll
ABI
B
gmethod
Application
Library 1.1
A
fmethod
hmethod
dll
ABI
B
gmethod
Application

Library 1.1. view of Class A's Method Table
0: fmethod
1: hmethod

Actual Method Table for an instance of B
0: fmethod
1: gmethod

The Problem


Library 1.0
dll
ABI
Application
Library 1.1
dll
ABI
Application

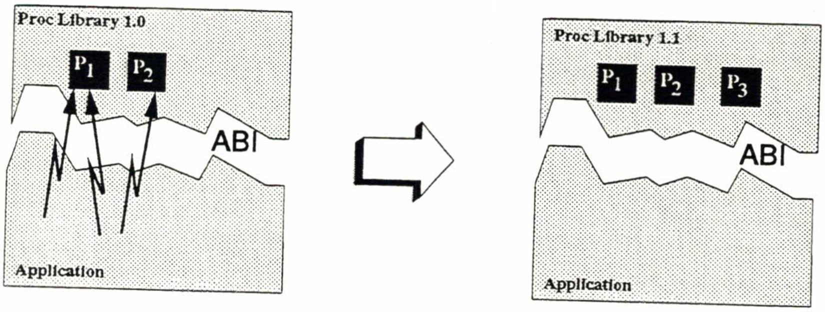
Solution:

Establish an **Engineering Discipline** with goal:

Only application alteration nessitates recompilation

Required: identify the **safe** transformations.

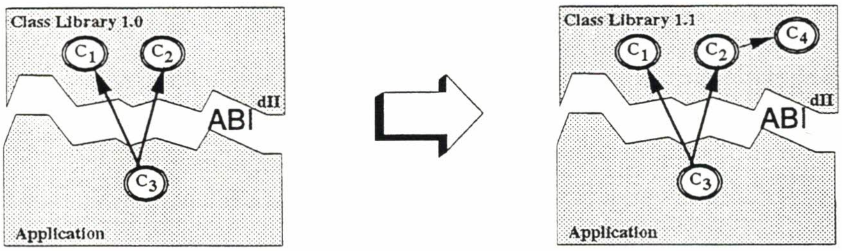
Procedural Programming


Proc Library 1.0
P_1
P_2
ABI
Application
Proc Library 1.1
P_1
P_2
P_3
ABI
Application

Expected safe transformations

0. Improvement of performance
1. Elimination of failures
2. Enlargement of the number of parameters
3. Addition of new procedures
4. Retraction of private procedures

Object Oriented Programming

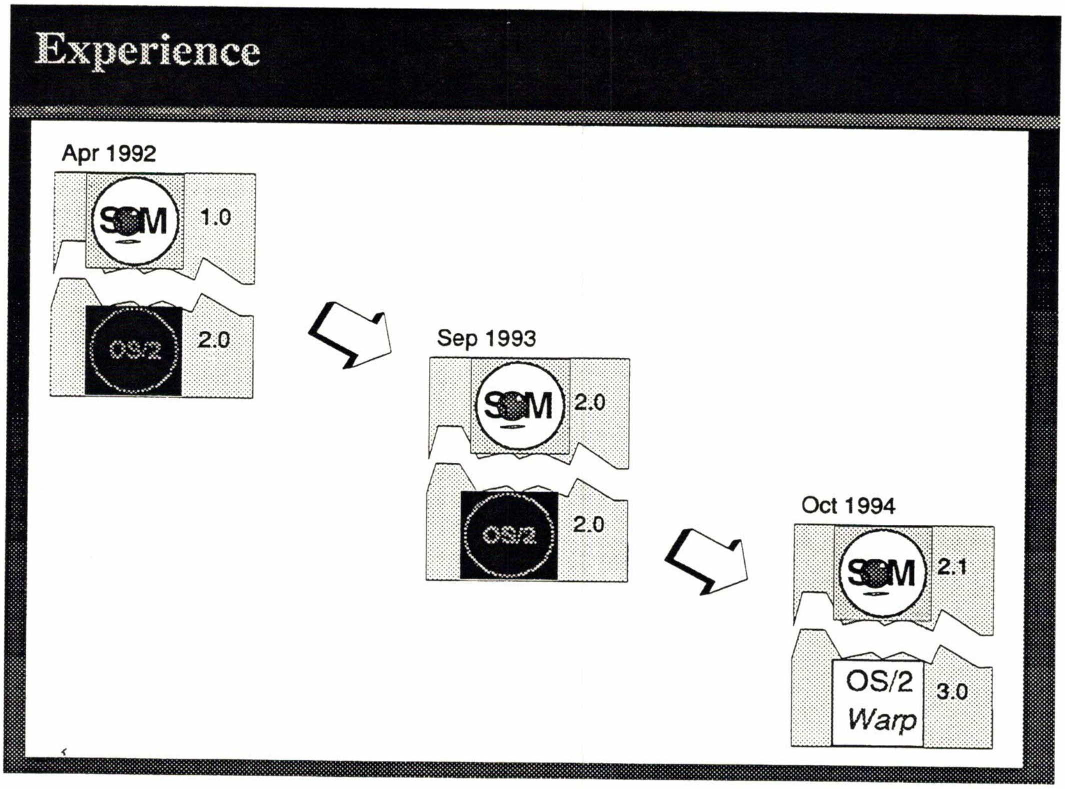


Expected safe transformations

5. Addition of new instance variable
6. Addition of new methods
7 Insertion of new classes
8. Migration of parent class downward
9. Migration of method upward
10. Retraction of private methods
11. Retraction of private instance data
12. Reordering of methods
13. Reordering of instance variables

SOM Delivers

	ST	C++	**SOM**	Delta /C++	Objective-C
0. Improve Performance	☑	☑	☑	☑	☑
1. Eliminate failures	☑	☑	☑	☑	☑
2. Enlarge parameter list	no	no	no	no	no
3. Add new procedures	n/a	☑	☑	☑	☑
4. Retract private procedures	n/a	☑	☑	☑	☑
5. Add new instance variables	no	no	☑	no	no
6. Add new methods	☑	no	☑	☑	☑
7. Insert new classes	☑	no	☑	?	no
8. Migrate parents downward	☑	no	☑	☑	no
9. Migrate methods upward	☑	no	☑	☑	☑
10. Retract private methods	n/a	no	☑	?	n/a
11. Retract private instance data	☑	no	☑	?	n/a
12. Reorder methods	☑	no	☑	☑	☑
13. Reorder instance variables	☑	no	☑	☑	no
14. Migrate metaclass constraint downward	no	no	☑	no	no



Experience

Apr 1992

SOM 1.0

OS/2 2.0

Sep 1993

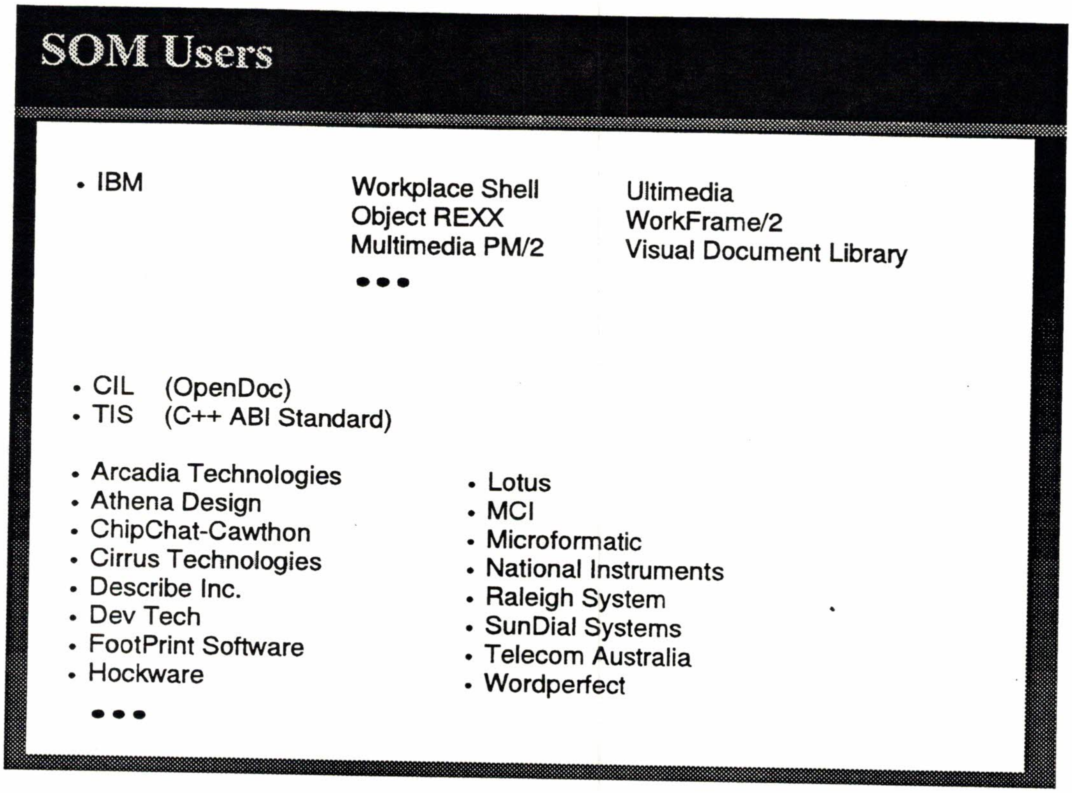
SOM 2.0

OS/2 2.0

Oct 1994

SOM 2.1

OS/2 *Warp* 3.0



SOM Users

- IBM

Workplace Shell
Object REXX
Multimedia PM/2
•••

Ultimedia
WorkFrame/2
Visual Document Library

- CIL (OpenDoc)
- TIS (C++ ABI Standard)

- Arcadia Technologies
- Athena Design
- ChipChat-Cawthon
- Cirrus Technologies
- Describe Inc.
- Dev Tech
- FootPrint Software
- Hockware

•••

- Lotus
- MCI
- Microformatic
- National Instruments
- Raleigh System
- SunDial Systems
- Telecom Australia
- Wordperfect

The Goals

- Language Neutrality
- State-of-the-Art Object-Oriented Capabliity
- Industrial Strength Class Libraries

are achieved

Making Reuse A Reality

SOM

Sales

1-800-3 IBM OS2

Service

somservice@austin.ibm.com

World Wide Web

http://www.austin.ibm.com/somservice

References

Scott Danforth, Paul Koenen, and Bruce Tate *Objects for OS/2* Van Nostrand Reinhold (1994).

Christina Lau *Object-Oriented Programming for SOM and DSOM* Van Nostrand Reinhold (1994).

Ira R. Forman, Scott H. Danforth, and Larry K. Raper "Support of Complied Class Libraries in SOM" *Proceedings of 7th IBM Conference on Object-Oriented Software Development* Santa Clara (July 11-15, 1994).

Ira R. Forman, Scott H. Danforth, and Hari H. Madduri "Composition of Before/After Metaclasses in SOM" OOPSLA '94 *Conference Proceedings* Portland (October 23-27, 1994).

Roger Sessions *Object Persistence* Prentice-Hall (expect: July 1995).

PRO 0408
Object REXX

Rick McGuire
IBM

Object REXX

Extending REXX for Tomorrow's Needs

Rick McGuire
Object REXX Development
IBM Endicott

Object REXX

- Work began in 1988
- Prototyped since 1989
- Beta version available on OS/2 Developers Connection Volume 6 (1-800-6DEVCON)
- Complete rewrite of interpreter
- Language architecture "in progress" and subject to change

REXX Uses Today

- Operating system command files (.CMD, EXEC, CLIST, etc.)
- Application macro language
- Personal productivity
- Tools
- Applications

Driving Forces

- People wish to use REXX for more, but...
  - Some REXX language features inhibit large application development (call-by-value, rules for data sharing)
  - Difficult to link REXX to some styles of interface (Presentation Manager, X-Windows)

Driving Forces

- Access to structured data is needed (records, arrays, other collections)
- Access to an underlying system object model is desired
  - ► *SOM*
  - ► *OpenDoc*
- Must keep REXX's easy-to-use nature

What is Object REXX?

- Object-oriented extensions to the REXX language
- Rich set of object classes
  - Structured data (arrays, records, dictionaries)
  - Visual/graphical classes (OS/2 Presentation Manager, Motif, etc.)
  - System specific objects (files, semaphores, queues)

What is Object REXX?

- Full System Object Model (SOM) interoperability
- IBM's scripting language for the Open Scripting Archtecture
- Frameworks for mobile computing

REXX Variables

- REXX variables are references to object instances
  - Variables can reference any object
  - Dynamic binding and class resolution
  - Automatic garbage collection
- Everything in REXX is an object

```
c = .animal~new("Pooh Bear")
```

Sending Messages

- Message send operations have three elements:
  - Target object
  - Method name
  - Optional arguments

```
c~setfood("Honey")
```

Object Methods

- Can be written in REXX or other languages
- Full access to object variables
- Passed message arguments

```
/* food = method for animal */
  expose food
  use arg newfood
  food = newfood
```

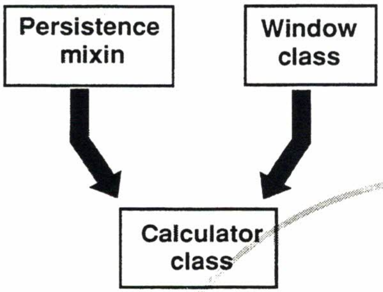
Inheritance

- Full facilities for class inheritance

```
::class dog subclass animal
::method bark
say "Unknown dog noise"
```

Multiple Inheritance

- Classes can inherit from multiple classes
- Abstract classes available via "mixins"

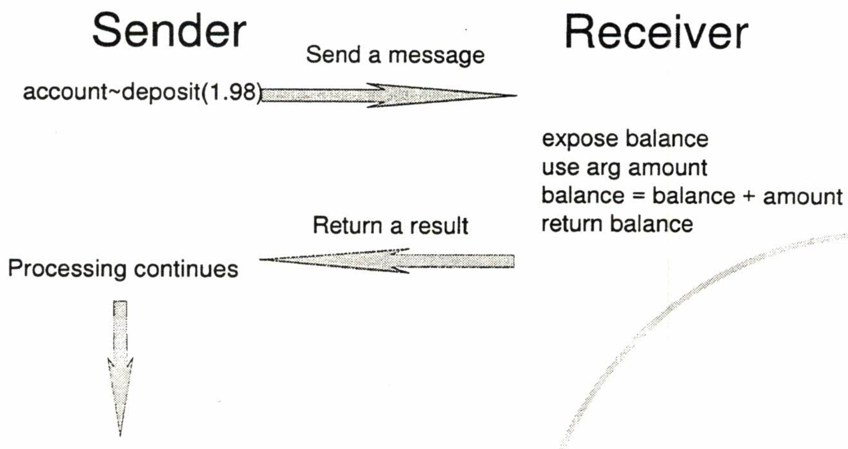


Persistence mixin

Window class

Calculator class

Sequential Execution



Sender

Receiver

Send a message

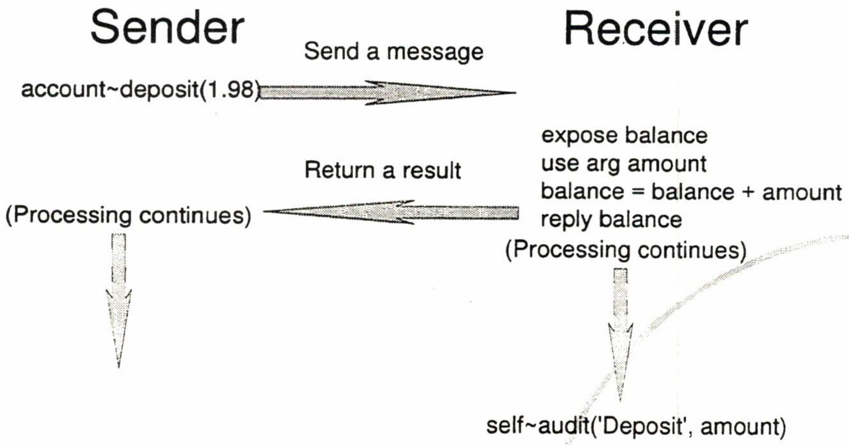
account~deposit(1.98)

expose balance
use arg amount
balance = balance + amount
return balance

Return a result

Processing continues

Concurrent Execution



Sender

Receiver

Send a message

account~deposit(1.98)

expose balance
use arg amount
balance = balance + amount
reply balance

Return a result

(Processing continues)

(Processing continues)

self~audit('Deposit', amount)

System Object Model Access

- Fully interoperable with the System Object Model (SOM)
  - Access non-REXX SOM objects from REXX
  - Access REXX objects from non-REXX languages
  - Subclass non-REXX SOM objects with REXX
  - Subclass REXX objects with non-REXX SOM code

Rexx Object

System Object Model Access

- First object-oriented language designed with SOM access in mind
  - SOM access is very natural extension of REXX object model
- Full access to any SOM objects, including Workplace Shell classes and OpenDoc parts
- SOM support includes access and interoperability with Distributed SOM

Rexx Object

Object REXX Benefits

- Upward compatible REXX language extensions
  - Existing programs continue to run without change
- Extends capabilities of applications using REXX as a macro language
- Removes barriers to large application development

Object REXX Benefits

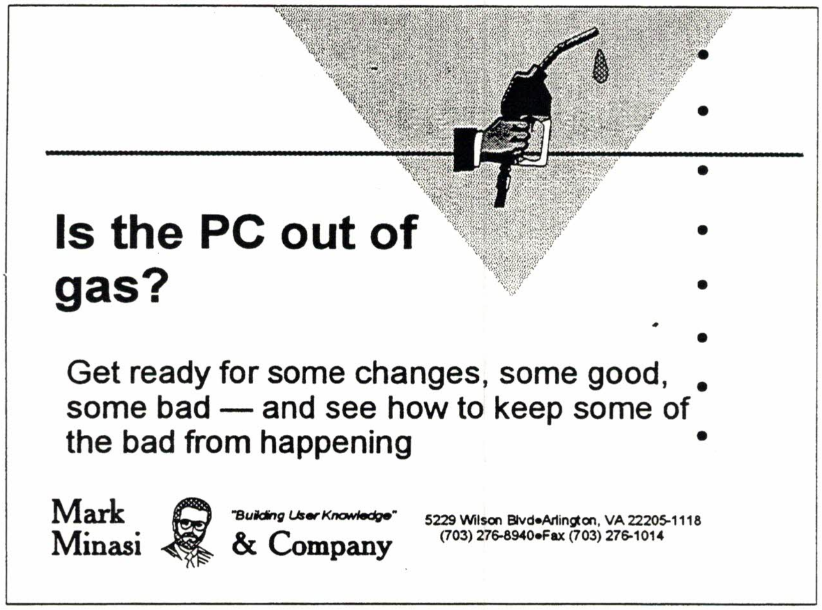
- Gives REXX users easy access to graphical user interfaces
- Acts as glue language to System Object Model applications
- Easiest way do develop Workplace Shell applications
- Excellent model for client-server computing

SYS 0402

Is the PC Out of Gas? What's New on the Desktop?

Mark Minasi

Mark Minasi & Company
Arlington, VA



Is the PC out of gas?

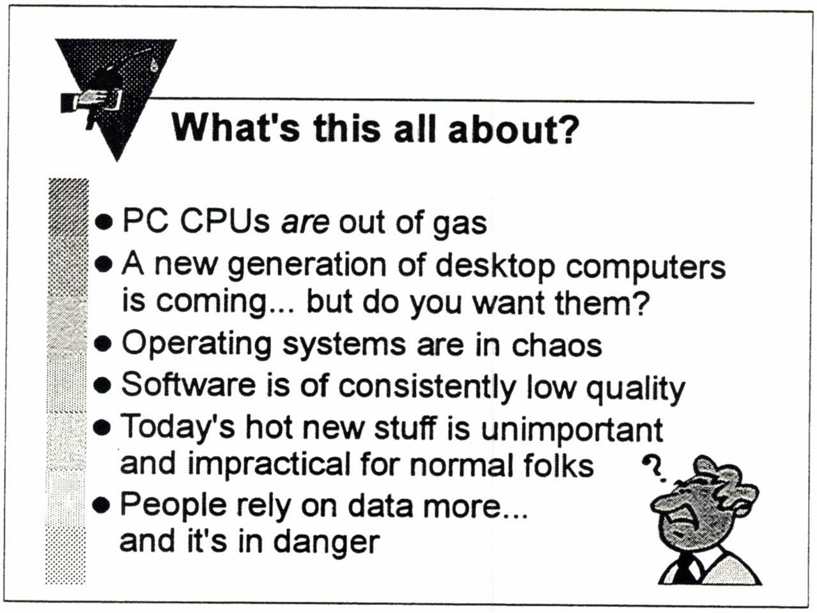
Get ready for some changes, some good, some bad — and see how to keep some of the bad from happening

Mark Minasi

"Building User Knowledge"

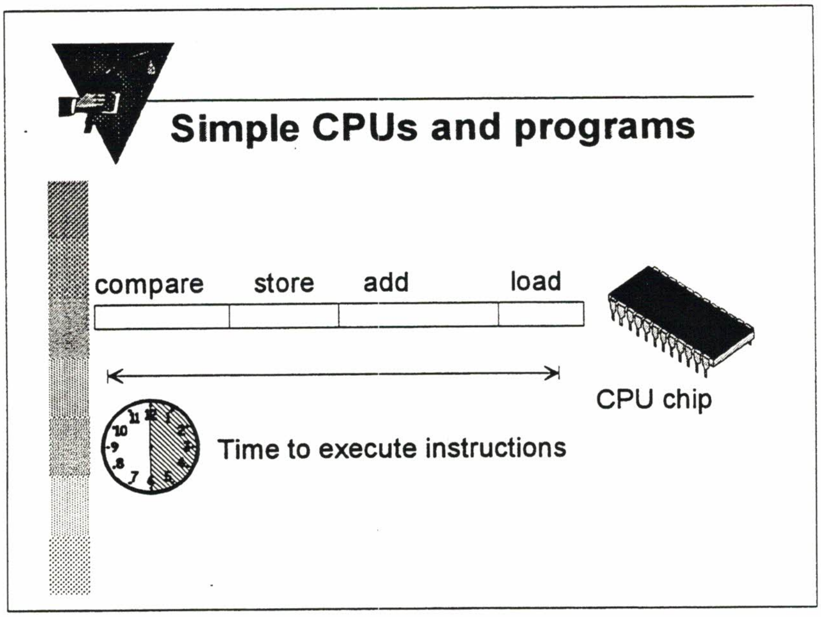
& Company

5229 Wilson Blvd•Arlington, VA 22205-1118
(703) 276-8940•Fax (703) 276-1014



What's this all about?

- PC CPUs *are* out of gas
- A new generation of desktop computers is coming... but do you want them?
- Operating systems are in chaos
- Software is of consistently low quality
- Today's hot new stuff is unimportant and impractical for normal folks
- People rely on data more... and it's in danger

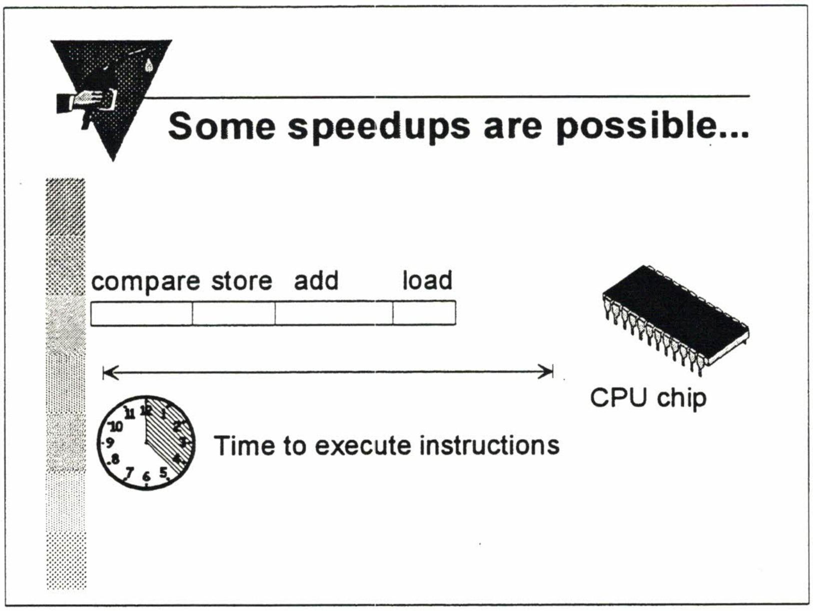


Simple CPUs and programs

compare store add load

CPU chip

Time to execute instructions



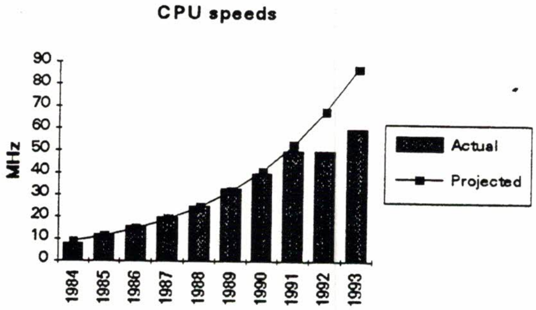
Some speedups are possible...

compare store add load

CPU chip

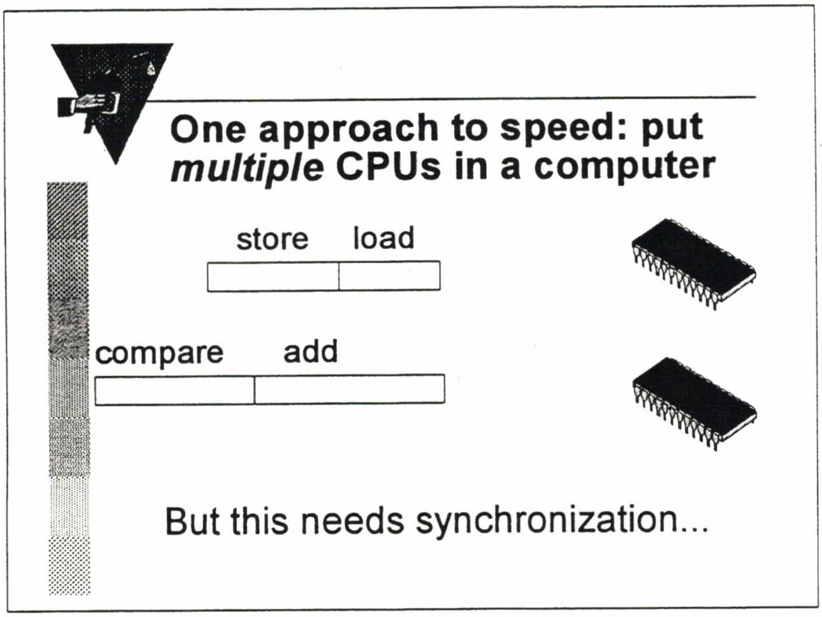
Time to execute instructions

But that way isn't working any more



Result: the PC hardware market is at a standstill

- Once, PCs cost $2K-$15K
- Now all cheaper than $5K
- The PC is becoming a commodity
- Less research, less innovation
- Manufacturers will have to differentiate
- Do we *care*? Do we really need more power?

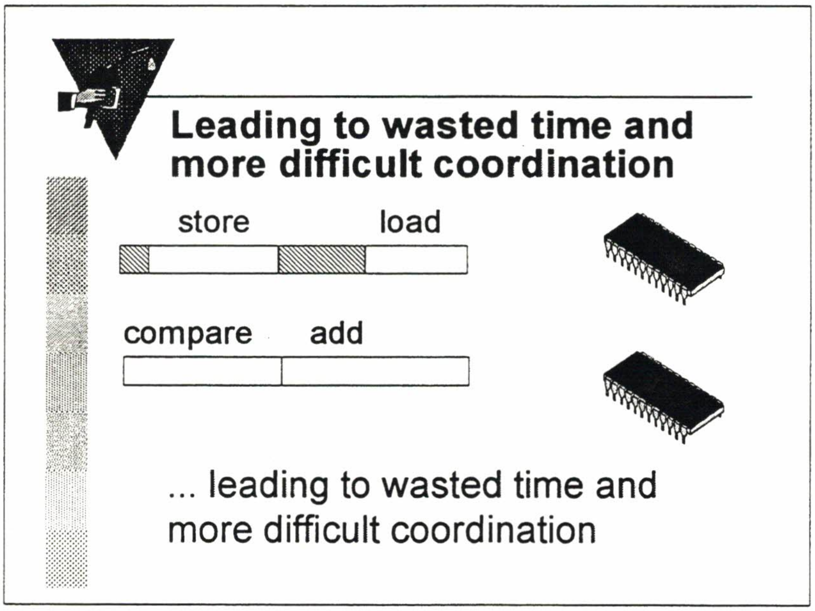


One approach to speed: put *multiple* CPUs in a computer

store load

compare add

But this needs synchronization...



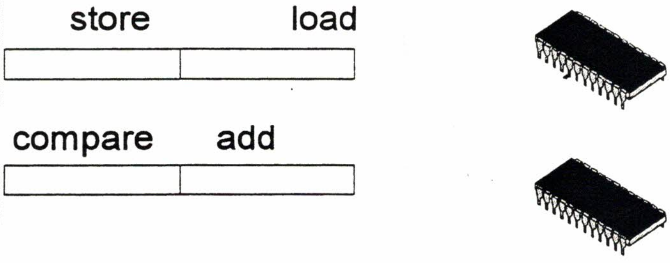
Leading to wasted time and more difficult coordination

store load

compare add

... leading to wasted time and more difficult coordination

Another answer: RISC (Reduced Instruction Set Chip)



store load

compare add

There *are*, of course, a few problems with RISC...

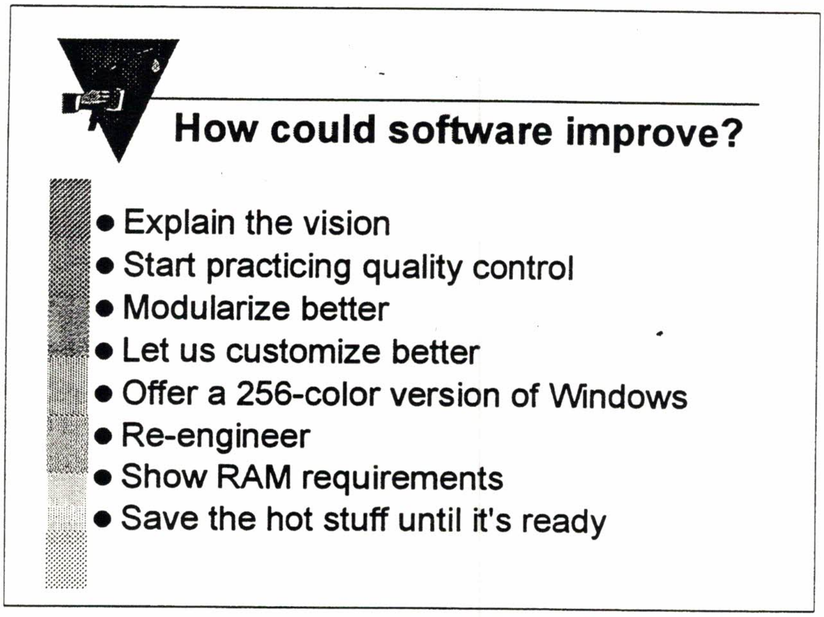
- It's incompatible with DOS
- It's more expensive: you can easily spend $10K-$15K on a RISC machine with
  - 16 MB RAM
  - 500 MB drive
  - LAN, SCSI connections
  - video accelerator
  - CD-ROM

How could hardware improve?

- Simplified PC design
- Computer education would be cheaper
- Do we need higher performance?
- You *needn't* replace something because it's "depreciated"
- Machines can be balanced better
- Add backup hardware as standard equipment

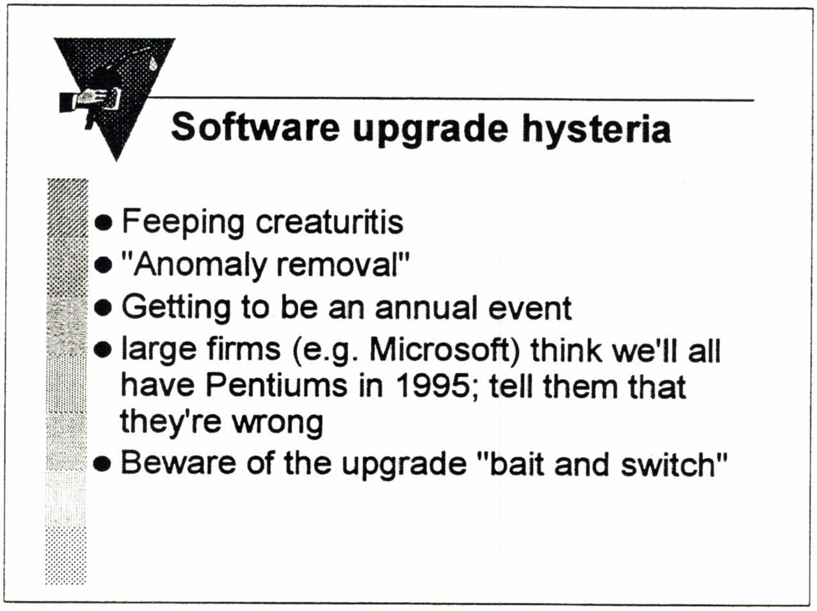
Software Issues

- People are moving to alternate OSes slowly
- Alternate OSes require heavy horsepower
- Users remain attached to DOS and DOS restrictions
- OSes don't support security well
- Windows weaknesses
- Suites are winning



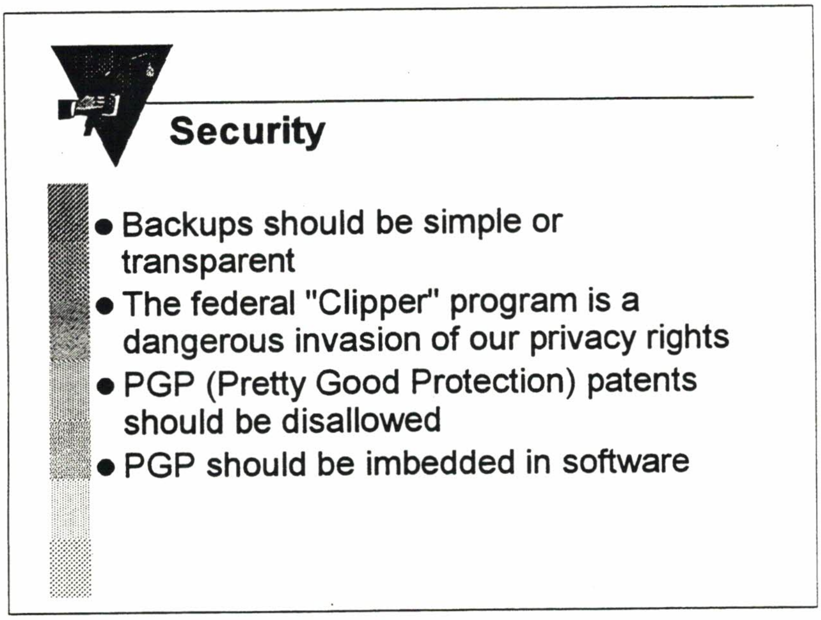
How could software improve?

- Explain the vision
- Start practicing quality control
- Modularize better
- Let us customize better
- Offer a 256-color version of Windows
- Re-engineer
- Show RAM requirements
- Save the hot stuff until it's ready



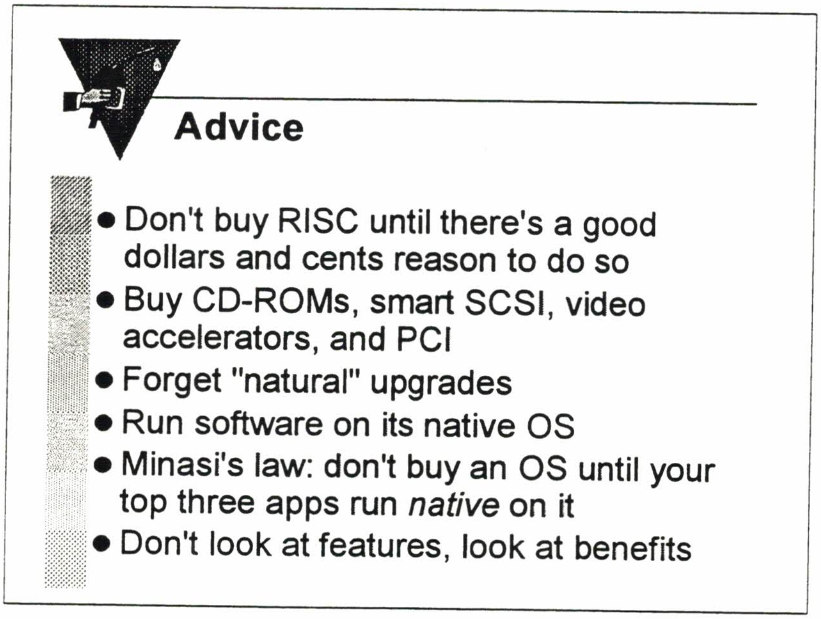
Software upgrade hysteria

- Feeping creaturitis
- "Anomaly removal"
- Getting to be an annual event
- large firms (e.g. Microsoft) think we'll all have Pentiums in 1995; tell them that they're wrong
- Beware of the upgrade "bait and switch"



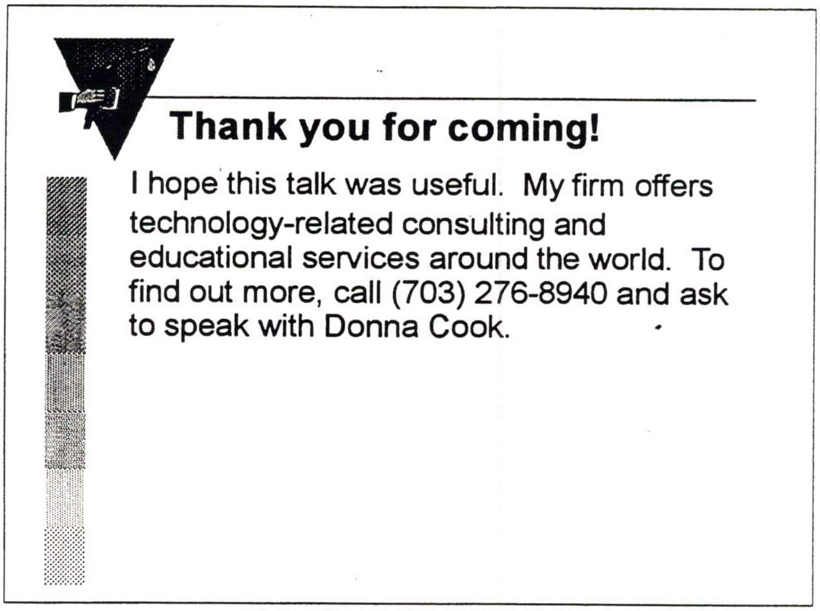
Security

- Backups should be simple or transparent
- The federal "Clipper" program is a dangerous invasion of our privacy rights
- PGP (Pretty Good Protection) patents should be disallowed
- PGP should be imbedded in software



Advice

- Don't buy RISC until there's a good dollars and cents reason to do so
- Buy CD-ROMs, smart SCSI, video accelerators, and PCI
- Forget "natural" upgrades
- Run software on its native OS
- Minasi's law: don't buy an OS until your top three apps run *native* on it
- Don't look at features, look at benefits



Thank you for coming!

I hope this talk was useful. My firm offers technology-related consulting and educational services around the world. To find out more, call (703) 276-8940 and ask to speak with Donna Cook.

SYS 0403

DB/2 and COBOL: Tools for Corporate Developers

Robert Chapman
Chapman & Associates
Aliso Viejo, Calif.

INTRODUCTION

The potential development environment for corporate programmers is so different today then it was four years ago that a complete review of the attributes of the two environments is a requirement for anyone wishing to get the most out of their development resources. This review is necessary for another reason also. Most companies today have invested heavily in workstation technology for their development personnel, but are getting very little return for the dollar on their investment. Bluntly stated, they have traded $10/month 3270 terminals for $150/month 3270 terminal emulators.

The main question that needs to be answered in this paper is twofold. First, why is development no cheaper and faster today then before the workstations were acquired. Second, given that the previous question accurately represents the state of data processing, what can we do to develop systems and applications cheaper and faster.

The resolution of this issue is investigated in the remainder of this paper by starting with a quick review of yesterday's environment, followed by a look at today's environment as it actually exists, and then a look at what it should be. Finally, the existing problems with applications development are discussed and solutions are proposed.

YESTERDAY'S DEVELOPMENT ENVIRONMENT

In the beginning there were punched cards, but we will not go back that far. Contemporary application development started with the advent of the 3270 display terminal (my apologies to non-IBM shops). This marked the start of direct man-machine interaction. Although Academics will certainly argue the historical accuracy of this statement I can confidently state that for the average data processing individual it is certainly true.

As the years went by more and more sophisticated development tools were added to the 3270 environment. Chief among these was the IBM Structured Programming Facility (ISPF or PDF) which provided a standard editing environment within the mainframe environment. This environment was extendable via the Dialog Manager facility and over the last 15 years additional tools, mostly editing type tools, were added to the ISPF environment. But the important point to be made is that even though ISPF has been extended in function it has not been extended in scope by either its users or its developers.

The major characteristics of this environment were as follows. Development activities took place on the 3270 terminal. This terminal, in its most common form, displayed 25 lines of 80 characters each, in a block transfer mode. Block transfer mode means that the information entered on the terminal is not sent to the computer until a specific operator request, such as striking the ENTER key, has been entered. The edit environment was supplied with a background job submission facility, along with tools allowing the developer to review the results of their background jobs without waiting for the physical printout. The background jobs executed using host files and databases. There were no WYSIWYG word processors and no spreadsheet programs available. Nor were any needed.

The direct consequences of this environment were four fold. First, application development and design were both host centric. Second, there were no GUI development tools or production systems. Third, programmer and designer productivity were improved over the card era of previous years, but still substantially lower than that needed by their user communities. Fourth, the applications coming out of data processing were still "low" in creativity. They were work horse applications, very few of which were sexy, in any sense of the word. Finally, there were very few fancy presentations.

TODAY'S DEVELOPMENT ENVIRONMENT - REAL LIFE

Today's environment is definitely a schizophrenic look at reality. Developers have massively powerful workstations residing on their desks, which in turn are connect to gigabytes of DASD via a LAN connection, which in turn is connected to a host system containing more MIPS then the most optimistic of us prognosticators of the future could ever imagine 15 years ago. But, the developers are still connected to the host via the venerable old 3270 interface, with multiple terminals being emulated instead of one single "real" 3270 terminal. They still use the ISPF editor. They still perform compiles in Batch. They still maintain all of their databases and files on the host. In fact the only thing that has changed in their lives is that their displays are now in color and they are all expected to perform word processing functions and utilize spreadsheet software to make their jobs easier.

COMPARING THE NEW TO THE OLD

In the old world developers interfaced to the host via a 3270 terminal, some with multisession capability. In the new world programmers interface with the host via emulated 3270 terminals sporting multisession capabilities. In the old world programmers compiled their programs in batch on the host system. In the new world programmers do the same. In the old world 3270 terminals cost about $10 per month to use. In the new world PC Workstations cost about $150 per month to use. In the old world programmers did not have to spend much of their working hours creating good word processing documents, or learning how to enter data into one or another of the spreadsheets on their workstation.

It seems pretty reasonable to me to say that the reason that productivity has not gone up, while costs have gone up, is easily related to the fact that the use of PC Workstations in the application development environment has only increased the workload. It has not reduced the work load on the individual programmer. In other words, development is spending more, and getting more work out of the programmers, but the work is not necessarily related to the job at hand.

TODAY'S DEVELOPMENT ENVIRONMENT - WHAT IT SHOULD BE

There are six things that today's development environment should be. First, it should be a GUI development environment. Second, it should have real time compilation. Third, all databases and files should be locally and natively accessible to the PC Workstation. Fourth, when an application could legitimately benefit from a GUI interface, it should get it. Fifth, all accesses to host databases or files should be transparent to both the application program and the development staff. Finally, it should be low cost.

The development environment should be GUI based because there are many different skills required during the application development process, and the use of a GUI environment would simplify the training needs of the majority of the staff. GUI's are great for high turnover and high complexity work environments.

Compilations should be performed on the Workstation, within the GUI environment. This gives the greatest speed and vastly improves the functionality of the editing process. Many GUI development environments allow the user to modify programs during the actual compilation processing, thus speeding application construction.

Host based databases and files often come with security restrictions wrapped around them. These restrictions, when combined with the bureaucratic processes needed to maintain the integrity of these resources, definitely have a costly effect on any development project. However, let it be said here that these same security requirements and paper work mills are absolutely needed, as otherwise many of the production files and physical resources would be rapidly corrupted and used up by the programming staff. This means that if the Workstations have the capacity, both CPU and DASD, to manage local databases and files, they should do so.

Finally, the application development environment must be low cost. It must be cheaper per unit of production than its predecessor environments.

Unfortunately for most applications developers their existing environment is none of the above.

PROBLEMS AND SOLUTIONS

Development is expensive. It is expensive on both the host and on the Workstation. But it is more expensive on the Workstation because many developers are using substandard languages like C, C++, Smalltalk, Easel, Applications Manager, etc. These languages are substandard for a variety of reasons. First, many of them are not full function. This means that they cannot perform all of the normal tasks which are necessary in a standard business application without resorting to calling modules written in different languages. Second, many of them are hopelessly complicated in both their structure, their assumptions, and their flexibility. The C language for instance is not complicated, but the syntax is highly cryptic, and not conducive to file handling (one reason for this is that file and I/O handling are not defined in the language itself, but included as a series of semi-standard function calls). The C++ language is hopelessly complicated. The novice C++ programmer must learn on the order of 400 different objects before they can operate on their own. While it is true that all of these objects add flexibility to the development environment, it also means that most people must spend years and years of their professional life becoming familiar with all of the different objects.

The simple solution to this problem of languages is to use COBOL on the Workstation. Today's COBOL is a superset of the features and functions of COBOL back in the 70's. The new features are so powerful that they rival the functionality of the so called "super" language of C. At the same time these features are so well integrated with the previous features of the language that it is a smooth and natural progression to their use.

Maintenance is expensive on the host, and on the Workstation. It is expensive on the host because host resources are expensive, and there is no GUI development environment. It is expensive on the Workstation because many of the systems running on the Workstation were developed in C, C++, Smalltalk, etc. Many of the applications running on these systems are virtually none maintainable because of the cryptic nature of the language.

The easy solution to this problem is to switch to COBOL for Workstation development. There are more COBOL programmers available in the world today than any other language. COBOL is easy to maintain (unless it is written like a C program). COBOL executes fast. And COBOL has built in access methods, which by itself should be justification enough for using the language.

Workstation development has also been hindered by the operating environment available to applications developers. On the PC the use of the DOS/Windows operating environments is a major deterrent to full utilization of the Workstation resources. DOS and Windows are highly susceptible to corruption by the applications executing under them. They are in fact, accidents waiting to happen, and any manager that places a business function handling the financials of the business on either of those environments should be fired.

Mac Workstations have a nice user interface, but up until now they have suffered from three major problem areas. First, cooperative multitasking is not robust enough for a business environment. Second, there are almost no development languages available for the Mac, other than C. Third, the non-PowerMacs have no parity memory, so there is no way a developer can be assured that the data entered by the user or system is actually the data being processed by the programs. I repeat my previous opinion on DOS/Windows - any manager that places a business function handling the financials of the business on this platform should be fired.

OS/2 is the solution to all of the above concerns. OS/2 provides a large memory space, multi-CPU support, multiprocessing support, multithread support, interprocess communications, LAN support, multilanguage support, industrial strength database managers and access methods, full host connect functionality, transparent and seamless host database and file access, a high degree of stability, and soon - cross platform compatibility.

WORKSTATION/HOST DATABASE COMPATIBILITY

The IBM DB2/2 product provides a high fit proper subset of DB2. This means that with very few exceptions you can migrate an application depending on a DB2 database to and from the Workstation. One of my clients has a 300 table DB2 database which I successfully implemented under DB2/2. It was necessary for me to change only two column names out of 2000 in order to make this migration.

The IBM DDCS/2 product allows the applications developer to make a seamless connection between applications running on the Workstation and databases residing on the host. This connection is fast enough so that the throughput constraints all lie on the Workstation side.

Access to both the local DB2/2 database and the host DB2 databases is available via three OS/2 facilities. Databases can be accessed via the OS/2 session command line commands. Thus allowing a user to connect to a database and solicit query responses and update table rows directly from the command line. Databases can be accessed from within a REXX procedure. Databases can be access from within an application program written in most of the different languages.

Finally, the DDCS/2 product allows applications to switch between host and local databases without program recompiles. This means that the interface is truly seamless, in every sense of the word.

On a related topic. The use of the IBM CICS product for OS/2 or the IBM Distributed Storage Manager product both allow the Workstation to get seamless access to any other type of host data file (Sequential, VSAM, PDS, Direct, etc.).

DB2/2 AND DDCS/2 PERFORMANCE

Both of these products have adequate performance today, and getting better as each release goes by. DDCS/2 has always had real good performance. I have never seen a Workstation that could process data faster than DDCS/2 could deliver it. DB2/2 is another story.

Relational database engines have a lot of work to perform even for the simplest of retrievals. This has often been a concern of both developers and designers. The original version of DB2/2, Extended Edition, was excruciatingly slow for most tasks. Extended Services, which was the original replacement for Extended Edition, was faster, but still inadequate for many business tasks. DB2/2 1.0 was the first that was fast enough to maintain a reasonable level of access throughput. The newer release of DB2/2 is probably faster still, and soon DB2/2 will support the decomposition of access requests so that different parts can be run on different processors at the same time. DB2/2 is very fast as relational engines go, and could be the fastest such engine on the market today for most business applications.

PROGRAMMER PRODUCTIVITY

The reason development still takes too long is that applications developers are not using their Workstations to the best advantage possible. Application development could take place on the Workstation, instead of through the Workstation, and this would increase productivity, and decrease development time.

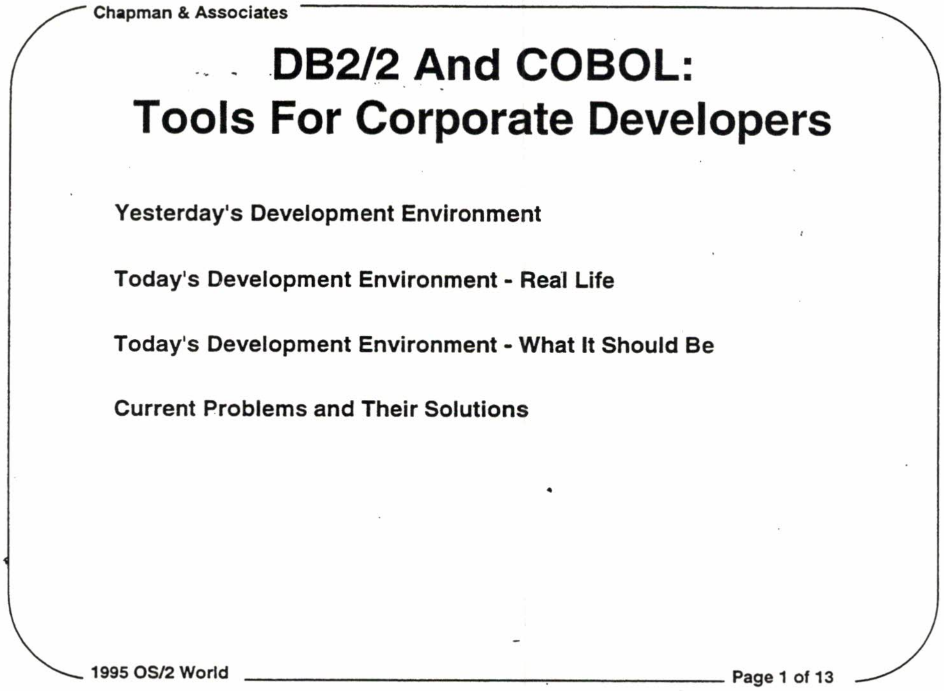
Even so, there are other factors involved in increasing the Workstation cost and decreasing the development flexibility.

The iterative development approach used by some installations is time consuming and wasteful. Can you imagine what D-Day would have been like if the Allies decided to first try to get a ship close to the beach, then two ships, then two ships and a troop transport, then ...? Developers should always find out what must be done before they go to do it.

Remove word processors, spread sheets, and presentation packages from the developers Workstation. Programmers should be programming, analysts should be analyzing, and designers should be designing. Once in a while someone needs to make a presentation, and specs always need to be written. But this widening of job responsibility is fast turning everyone into someone who can do nothing.

Complete your application design before you start writing program specifications. Complete the program specifications before you begin coding. Complete your test data and test cases before you begin testing.

The single biggest problem with applications development is a lack of discipline, a missing work ethic so to speak. Do not bring this sloppy development ethic to the Workstation as their are fewer management tools available to track and rectify these situations



Chapman & Associates

DB2/2 And COBOL:
Tools For Corporate Developers

Yesterday's Development Environment

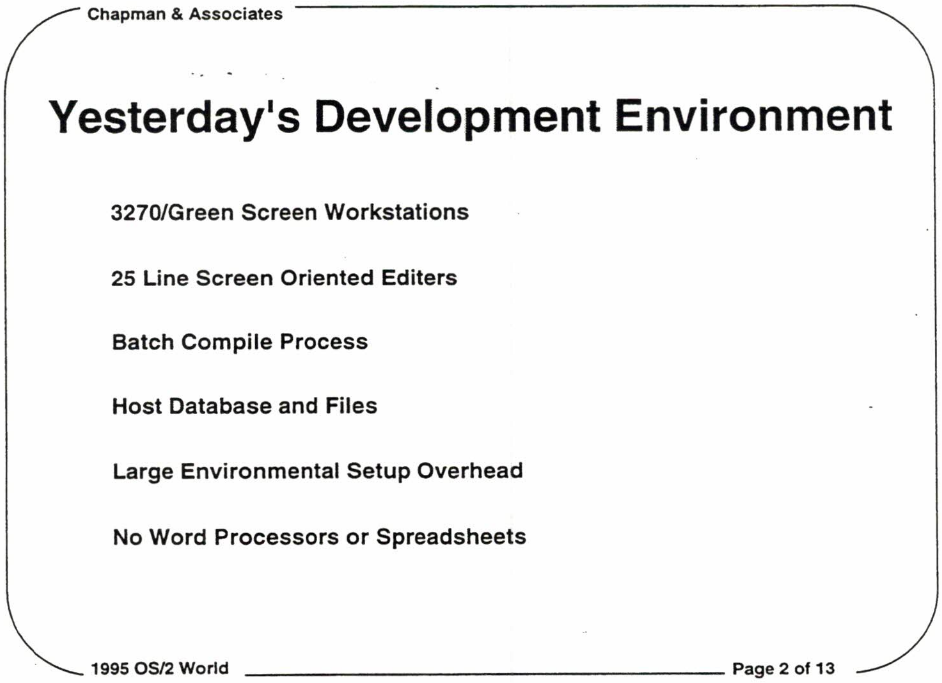
Today's Development Environment - Real Life

Today's Development Environment - What It Should Be

Current Problems and Their Solutions

1995 OS/2 World

Page 1 of 13



Chapman & Associates

Yesterday's Development Environment

3270/Green Screen Workstations

25 Line Screen Oriented Editers

Batch Compile Process

Host Database and Files

Large Environmental Setup Overhead

No Word Processors or Spreadsheets

1995 OS/2 World

Page 2 of 13

Yesterday's Consequences

Host Centric Application Design

Non GUI Development/Production Systems

Low Creativity

Very Few Fancy Presentations

Today's Development Environment Real Life

High Speed, Multi-Tooled PC/MAC Workstation

Multiple Session 3270 Emulation

25 Line Screen Oriented Editers

Batch Compile Process

Host Database and Files

Large Environmental Setup Overhead

Lots of Word Processing and Spreadsheet Manipulations

Today's Consequences

Host Centric Application Design

Non GUI Development/Production Systems

Low Creativity

Lots Of Fancy Presentations

Compare The Old To The New

First - Development Is Done On 3270 Emulators

Second - Host Compiles And Testing Are The Norm

Third - Multiple 3270 Sessions Are Available In Old And New

Fourth - Workstation Cost For New Is 10 Times That Of Old

Fifth - Everyone Gets To Do Word Processing With New

Today's Development Environment
What It Should Be

GUI Development Environment

Real Time Compilation

Local Database And Files

GUI Applications - When Appropriate

Transparent Host Database And File Access

Low Cost

Current Problems And Solutions

High Cost Development
    COBOL Is The Solution

High Cost Maintenance
    COBOL Is The Solution

Workstation Development Environment
    OS/2 Is The Solution

Workstation/Host Database Compatibility
    DB2/2 Is The Solution

Programmer Productivity
    Dump The Word Processing And Spreadsheets

High Cost Development/Maintenance

More COBOL Programmers Available Than Any Other Language

More Business Experience In COBOL Than Any Other Language

More COBOL Programs Than Any Other Language

GUI Applications Are Easier To Develop In COBOL (Professional Experience)

COBOL Executes Just As Fast And Is Easier To Debug

COBOL Is More Transportable Than Any Other Language

Workstation Development Environment

OS/2 Provides
- Large Memory Space
- Multi-CPU Support (SMP)
- Multi-Processing Support
- Multi-tasking (Threads) Support
- Interprocess Communications Support
- LAN Support
- Language Support
- Industrial Strength Database Managers (DB2/2, Oracle, Sybase)
- Stability
- Soon To Come - Cross Platform Compatibility

WS/Host Database Compatibility

DB2/2 Provides A Proper Subset Of DB2

DDCS/2 Provides A Seamless Connection From WS To Host

DB2/2 Process Is Available Via OS/2 Command Line Commands

DB2/2 Process Is Available Via OS/2 REXX Procedures

DB2/2 Process Is Available Via COBOL/C/C++/... Languages

DDCS/2 Allows Command Line, REXX Procedures And Languages To Access Host Databases Without Change

---

DB2/2 And DDCS/2 Performance

PC Platforms Are Not As Powerful As Mainframes

PC Platforms Have From 1/1000 to 1/10000 The Power

DB2/2 Will Perform 10 to 20 Accesses Per Second

New DB2/2 Versions Will Provide Multiple Engine Support

DDCS/2 Performance Depends On Host Connect Link - But It Can Provide Data Faster Than The PC Hard Disk Subsystem

Programmer Productivity

Dump The Iterative Development Approach

Remove Word Processor And Spreadsheet Programs

Complete Design Before Specifications Are Written

Complete Specifications Before Coding Starts

Define And Build Test Data And Test Cases Before Testing Starts

Discipline, Discipline, And Discipline

SYS 0405

LAN Server Administration Tips and Techniques

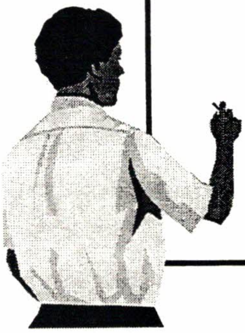
Gary Hunt
IBM

LAN SERVER 4.0
ADMINISTRATOR
TIPS AND TECHNIQUES

LAN SYSTEMS DEVELOPMENT
IBM AUSTIN

TIPS AND TECHNIQUES

TOPICS



- ARRANGING SERVERS
- SHARING RESOURCES
- CONTROLLING ACCESS
- MANAGING RESOURCES
- TUNING FOR PERFORMANCE

IBM LAN Systems

ARRANGING SERVERS

SERVER ROLES

DOMAIN CONTROLLER
  HOME OF THE DOMAIN CONTROL DATABASE
  (EVERY NETWORK'S GOTTA HAVE ONE)
  DOMAIN AUTHENTICATION AND LOGON ASSIGNMENTS
  ROLE: PRIMARY

ADDITIONAL SERVERS
  ROLE: MEMBER OR BACKUP

IBM LAN Systems

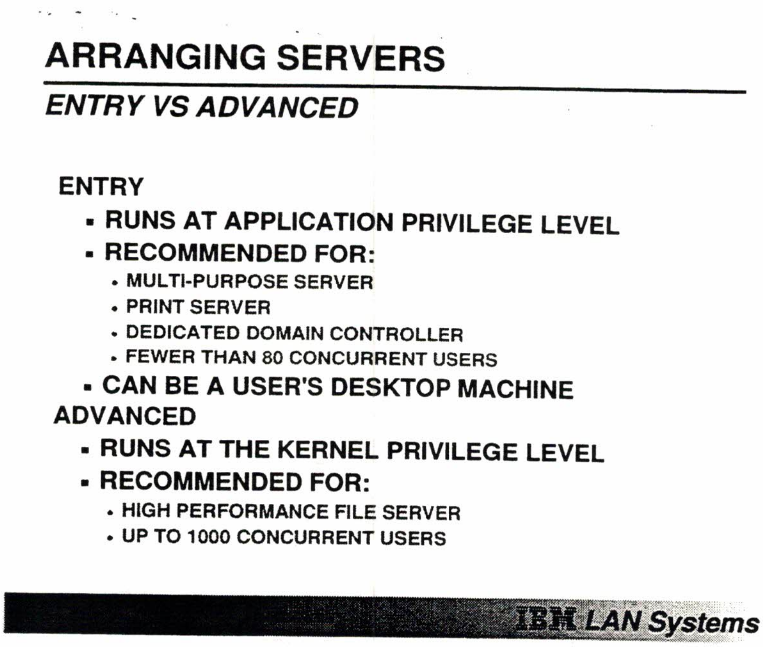
ARRANGING SERVERS

SPECIALIZED VERSUS MULTI-PURPOSE

SPECIALIZED SERVERS...
  ALLOW FOR OPTIMUM PERFORMANCE TUNING
  MINIMIZE DUPLICATION OF APPLICATIONS AND DATA
  FACILITATE RESOURCE POOLING
  FACILITATE EFFICIENT BACKUP/RECOVERY STRATEGIES

MULTI-PURPOSE SERVERS...
  PROVIDE FLEXIBILITY
  LOCAL SUPPORT FOR SMALL GROUPS

IBM LAN Systems



ARRANGING SERVERS

ENTRY VS ADVANCED

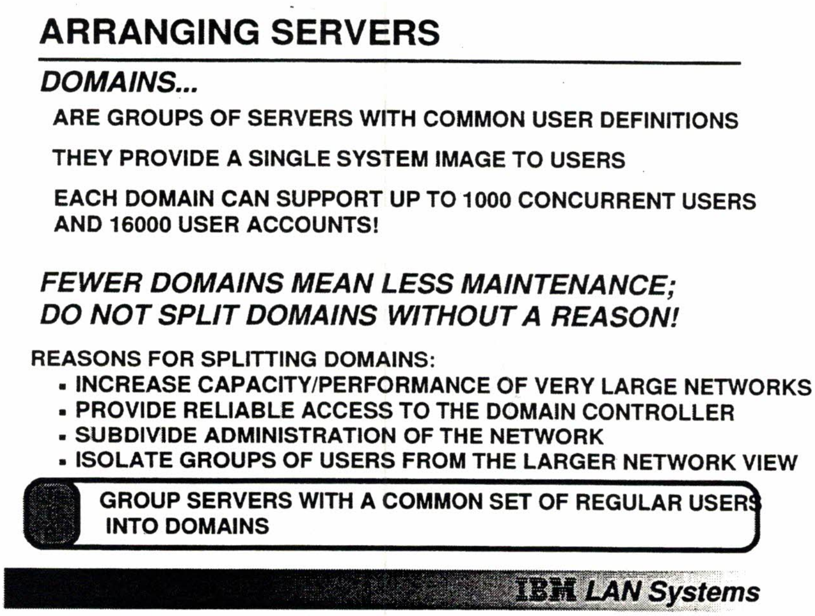
ENTRY

- **RUNS AT APPLICATION PRIVILEGE LEVEL**
- **RECOMMENDED FOR:**
  - MULTI-PURPOSE SERVER
  - PRINT SERVER
  - DEDICATED DOMAIN CONTROLLER
  - FEWER THAN 80 CONCURRENT USERS
- **CAN BE A USER'S DESKTOP MACHINE**

ADVANCED

- **RUNS AT THE KERNEL PRIVILEGE LEVEL**
- **RECOMMENDED FOR:**
  - HIGH PERFORMANCE FILE SERVER
  - UP TO 1000 CONCURRENT USERS

IBM LAN Systems



ARRANGING SERVERS

DOMAINS...

ARE GROUPS OF SERVERS WITH COMMON USER DEFINITIONS

THEY PROVIDE A SINGLE SYSTEM IMAGE TO USERS

EACH DOMAIN CAN SUPPORT UP TO 1000 CONCURRENT USERS AND 16000 USER ACCOUNTS!

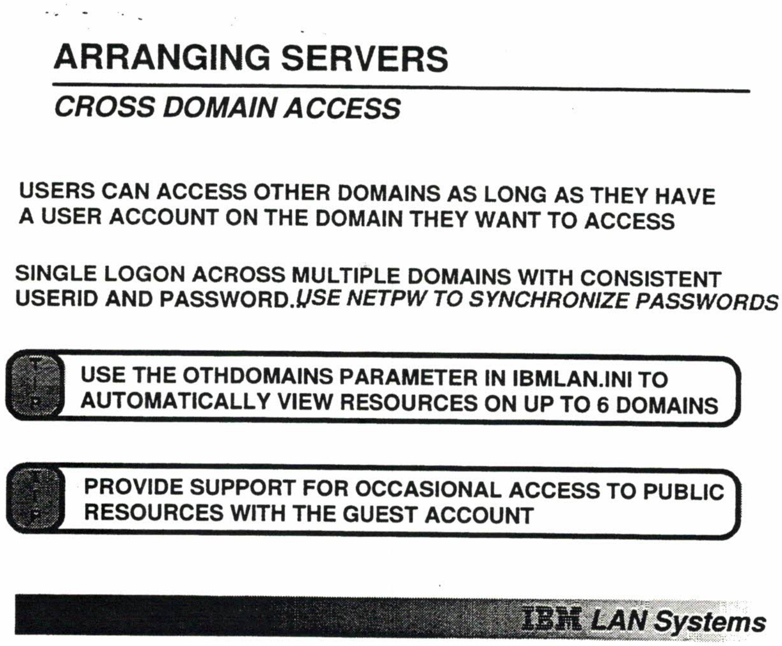
FEWER DOMAINS MEAN LESS MAINTENANCE;
DO NOT SPLIT DOMAINS WITHOUT A REASON!

REASONS FOR SPLITTING DOMAINS:

- INCREASE CAPACITY/PERFORMANCE OF VERY LARGE NETWORKS
- PROVIDE RELIABLE ACCESS TO THE DOMAIN CONTROLLER
- SUBDIVIDE ADMINISTRATION OF THE NETWORK
- ISOLATE GROUPS OF USERS FROM THE LARGER NETWORK VIEW

GROUP SERVERS WITH A COMMON SET OF REGULAR USERS INTO DOMAINS

IBM LAN Systems



ARRANGING SERVERS

CROSS DOMAIN ACCESS

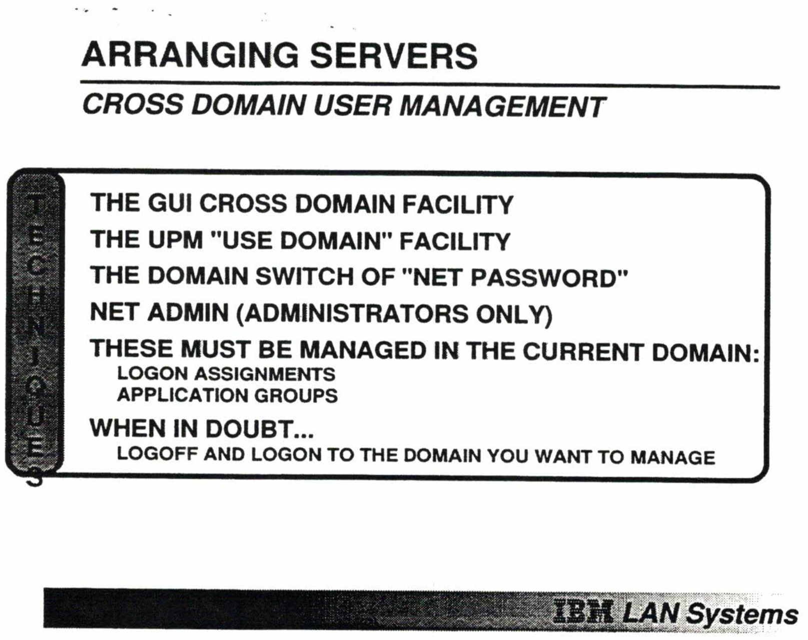
USERS CAN ACCESS OTHER DOMAINS AS LONG AS THEY HAVE A USER ACCOUNT ON THE DOMAIN THEY WANT TO ACCESS

SINGLE LOGON ACROSS MULTIPLE DOMAINS WITH CONSISTENT USERID AND PASSWORD.*USE NETPW TO SYNCHRONIZE PASSWORDS*

TIP USE THE OTHDOMAINS PARAMETER IN IBMLAN.INI TO AUTOMATICALLY VIEW RESOURCES ON UP TO 6 DOMAINS

TIP PROVIDE SUPPORT FOR OCCASIONAL ACCESS TO PUBLIC RESOURCES WITH THE GUEST ACCOUNT

IBM *LAN Systems*



ARRANGING SERVERS

CROSS DOMAIN USER MANAGEMENT

TECHNIQUES

- THE GUI CROSS DOMAIN FACILITY
- THE UPM "USE DOMAIN" FACILITY
- THE DOMAIN SWITCH OF "NET PASSWORD"
- NET ADMIN (ADMINISTRATORS ONLY)
- THESE MUST BE MANAGED IN THE CURRENT DOMAIN:
  - LOGON ASSIGNMENTS
  - APPLICATION GROUPS
- WHEN IN DOUBT...
  - LOGOFF AND LOGON TO THE DOMAIN YOU WANT TO MANAGE

IBM *LAN Systems*

SHARING RESOURCES

HOW ARE RESOURCES SHARED?

TECHNIQUES

ADMINISTARTIVE GUI
- AT SERVER STARTUP
- AS REQUIRED BY USER
- MANUALLY START AND STOP SHARE

DOS OR OS/2 COMMAND LINE
- NET SHARE

APPLICATION PROGRAMMING INTERFACE
- NetShareAdd

NOTE

RESOURCES MUST ALREADY BE SHARED TO BE ACCESSED FROM OUTSIDE THEIR DOMAIN

IBM *LAN Systems*

SHARING RESOURCES

WHAT ARE ALIASES?

ALIASES ARE THE LAN SERVER DIRECTORY TECHNOLOGY
- SINGLE SYSTEM VIEW TO RESOURCES ON MULTIPLE SERVERS
- LOCATION INDEPENDANCE
- RESOLVES TO \\SERVERNAME\NETNAME
- DEFINE SHARE CONFIGURATION FOR ALL SERVERS IN A DOMAIN
- DEFINE USER LOGON CONNECTIONS
- SIMPLIFY BROWSE AND CONNECT

TIP

DEFINE ALIASES FOR RESOURCES TO SIMPLIFY USER ACCESS AND FUTURE MAINTENANCE

IBM *LAN Systems*

SHARING RESOURCES

BEYOND FILES AND PRINTERS

PUBLIC AND PRIVATE APPLICATIONS

- OS/2, DOS AND WINDOWS APPLICATIONS SUPPORTED
- ENVIRONMENT CONFIGURED VIA GUI OR NET APP COMMAND

> REMEMBER TO PROVIDE WRITE ACCESS TO ANY WORK AND DATA DIRECTORIES; ALSO NEEDED FOR ICON USAGE

REMOTE IPL

- OS/2 AND DOS SUPPORTED
- NUMEROUS UTILITIES AVAILABLE FOR MANAGING REMOTE IPL

> REMOTE IPL PUTS A HIGH DEMAND ON SERVER CPU AND STORAGE...FOR BEST RESULTS, FOLLOW TUNING ADVICE GIVEN IN THE NETWORK ADMINISTRATOR REFERENCES.

IBM *LAN Systems*

CONTROLLING ACCESS

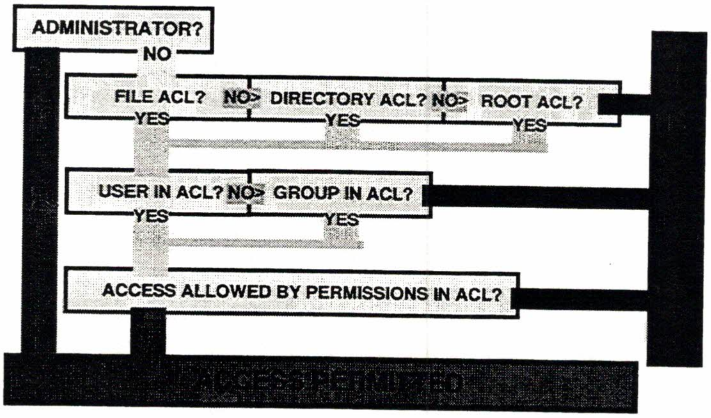
3 STAGES TO ACCESSING A RESOURCE

1) **LOGGING ON**
   REGISTERS A USER ID AND PASSWORD IN THE LOCAL CLIENT FOR USE IN SUBSEQUENT RESOURCE REQUESTS
2) **CONNECTING TO A RESOURCE ("NET USE")**
   ALLOWS A CLIENT TO ESTABLISH A NETBIOS SESSION WITH THE SERVER
   GRANTS ACCESS TO NETWORK; NOT TO RESOURCES!
3) **ACCESSING A RESOURCE**
   CONTROL IS ENFORCED WHEN AN ATTEMPT IS MADE TO OPEN A SHARED RESOURCE.

IBM *LAN Systems*

CONTROLLING ACCESS

FLOWCHART



ADMINISTRATOR?

NO

FILE ACL? NO> DIRECTORY ACL? NO> ROOT ACL?

YES YES YES

USER IN ACL? NO> GROUP IN ACL?

YES YES

ACCESS ALLOWED BY PERMISSIONS IN ACL?

ACCESS PERMITTED

IBM LAN Systems

CONTROLLING ACCESS

TIPS FOR MANAGING ACCESS CONTROL

TIPS

USE THE PARTITION ACL (C:) INSTEAD OF APPLYING THE ACL ON THE ROOT (C:\)
- CREATES A DEFAULT ACL FOR SUBDIRECTORIES IN WHICH THE ACL IS NOT EXPLICITLY DEFINED; DOES NOT OVERWRITE EXISTING ACLS

TIPS

USE THE NEW ADMINISTRATIVE GUI AND ACCESS CONTROL MANAGER (ACM) APPLET
- EASILY VIEW, CREATE, COPY, CHANGE, MOVE AND DELETE ACCESS CONTROL LISTS

IBM LAN Systems

CONTROLLING ACCESS

MORE TIPS

TIPS

CREATE GROUPS THAT CORRESPOND TO ACCESS CONTROL POLICIES RATHER THAN DEPARTMENTS
- EXAMPLES: READ/EXECUTE, READ/WRITE/CREATE
- LESS MAINTENANCE DUE TO REORGANIZATIONS

FEWER GROUP ACCOUNTS NEEDED

TIPS

GRANT ACCESS PRIVILEGE TO USERS BY ADDING THEM TO GROUPS

IBM *LAN Systems*

MANAGING RESOURCES

BACKUP AND RECOVERY

TIPS

DON'T WAIT FOR AN EMERGENCY TO FORMULATE A BACKUP AND RECOVERY PLAN!

TECHNIQUES

UNINTERRUPTABLE POWER SUPPLY

FOR THE DOMAIN CONTROL DATABASE
- SETUP A BACKUP DOMAIN CONTROLLER
- SETUP DCDB REPLICATION
- NEW APPLETS FOR IMPORTING AND EXPORTING DCDB TO A DATABASE

FOR DATA
- USE MIRRORING AND DUPLEXING
- USE "AT" COMMAND TO SCHEDULE REGULAR BACKUPS
- USE 3RD PARTY RAID OPTIONS

IBM *LAN Systems*

MANAGING RESOURCES

ROUTINE MAINTENANCE

TIPS

USE OPERATOR PRIVILEGE TO DELEGATE ROUTINE TASKS SUCH AS MANAGING PRINT QUEUES AND EXISTING USER ACCOUNTS

TECHNIQUES

FOR AUTOMATING ROUTINE MAINTENANCE...
- COMMAND AND BATCH FILES
- NEW LAN REXX EXTENSIONS AND UTILITIES
- APPLICATION PROGRAMMING INTERFACE
- SCHEDULING VIA THE "AT" COMMAND

FOR UPGRADING SOFTWARE ACROSS THE NETWORK
- CONFIGURATION/INSTALLATION/DISTRIBUTION (CID)
- NETDOOR

IBM LAN Systems

MANAGING RESOURCES

MONITORING RESOURCE USAGE

TECHNIQUES

SERVER CAPACITY VERSUS USAGE
- HARD DISK SPACE...NOW ENFORCEABLE LIMITS!
- CPU UTILIZATION...OS/2 PULSE, SPM/2, NETVIEW MONITOR
- MEMORY (WORKING SET)...SPM/2, NETVIEW MONITOR
- HPFS CACHE...CACHE386 UTILITY (ADVANCED SERVER)
- BUFFERS...NET ERRORS, NET STATISTICS, SPM/2

SECURITY/METERING
- UNSUCCESSFUL LOGON ATTEMPTS...AUDIT
- ATTEMPTS TO ACCESS PROTECTED SERVICES...AUDIT
- ACL AND USER ACCOUNT CHANGES...AUDIT
- APPLICATION USAGE...AUDIT, VENDOR, NETDOOR

PERFORMANCE
- THROUGHPUT AND RESPONSE...SPM/2, NETVIEW MONITOR, WATCHIT, BENCHMARKS (COMMERCIAL OR "ROLL YOUR OWN")

IBM LAN Systems

TUNING FOR PERFORMANCE

TOP TIPS FOR SERVERS

TIPS

1) USE LAN TUNE FOR INITIAL CONFIGURATION
   RERUN TO TUNE AFTER MAJOR NETWORK CHANGES

2) USE ADVANCED VERSION OF LAN SERVER FOR OPTIMUM FILE ACCESS WITH MORE THAN 80 USERS
   PLACE ALL SHARED FILES ON AN HPFS PARTITION

3) PROVIDE SUFFICIENT MEMORY FOR WORKING SET

4) ALLOCATE SUFFICIENT CACHE
   /USEALLMEM IF ADAPTER SUPPORTS USE OF MEMORY ABOVE THE 16 MBYTE ADDRESS

5) PROVIDE ADEQUATE CPU; MONITOR UTILIZATION

6) USE ADDITIONAL NICS IF CPU IS ADEQUATE BUT A BOTTLENECK IN NIC THROUGHPUT IS DETECTED

7) USE BUSMASTER NICS TO OFFLOAD CPU AS WELL AS IMPROVE THROUGHPUT THROUGH THE NIC

IBM *LAN Systems*

TUNING FOR PERFORMANCE

TIPS FOR SPEEDING UP LOGON

IF NETWORK IS FAST AND RELIABLE...

REDUCE NETBIOSRETRIES FROM 8 TO 1

SAVES 3.5 SECONDS PER NETBIOS NAME ADDED

OTHER TECHNIQUES FOR SPEEDING UP LOGON

DON'T RUN THE MESSENGER SERVICE IF NOT USED

SET MULTILOGON TO YES

CONFIGURE A BACKUP DOMAIN CONTROLLER

CHANGE LOGON FROM DOMAIN TO LOCAL

IBM *LAN Systems*

REFERENCES

OS/2 LAN SERVER NETWORK ADMINISTRATORS REFERENCE - VOLUMES 1, 2 AND 3

EXPERIENCES WITH THE IBM OS/2 LAN SERVER VERSION 3.0 NEW FUNCTIONS - IBM ITSC (GG24-3959-01)

IBM OS/2 LAN SERVER 3.0 SYSTEM RECOVERY CONSIDERATIONS - IBM ITSC (GG24-4043-00)

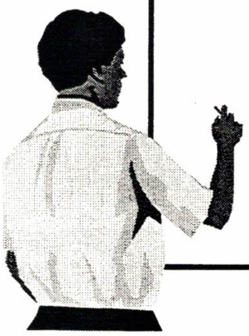
AUTOMATED INSTALLATION FOR CID ENABLED OS/2 V2.X - IBM ITSC (GG24-3783-01)

LAN SERVER ADMINISTRATION 2 CERTIFICATION REFERENCE

IBM *LAN Systems*

TIPS AND TECHNIQUES

TOPICS



- ARRANGING SERVERS
- SHARING RESOURCES
- CONTROLLING ACCESS
- MANAGING RESOURCES
- TUNING FOR PERFORMANCE

IBM *LAN Systems*

CS 0507

Basic Client/Server Programming with CPI-C

Dave Lutz
IBM

Agenda

- Level-set on terms
- Why use CPI-C?
- CPI-C starter set calls
- Classic transactions
- Sample code

APPC Verbs versus CPI-C Calls

★ *APPC interface varies across platforms.*

OS/2

opcode
ext_code
rc
sec_rc
tp_id
conv_id
rts_rcvd
data_length
data
send_type

vcb

```
APPC( vcb)
```

AS/400

```
_Rwrite( icffptr,
         data,
         data_length )
```

MVS (APPC/MVS)

```
CALL ATBSEND(
          conv_id,
          send_type,
          data_length,
          access_token,
          data,
          rts_rcvd,
          notify_type,
          rc )
```

APPC Verbs versus CPI-C Calls

★ *CPI-C calls are consistent on all platforms!*

```
cmsend(                  /* Send_Data              */
     conversation_ID,   /* I: conversation ID     */
     data_buffer,       /* I: send this buffer    */
     send_length,       /* I: length to send      */
     control_info_rcvd, /* O: control information */
     return_code);      /* O: return code         */
```

(C) IBM Corporation 1995

Conversation States

- CPI-C calls can only be issued in specific states
- Primary CPI-C states:
  - Reset
  - Initialize
  - Send
  - Receive
  - Confirm
- Some calls cause state changes
- Full state table is provided in CPI-C reference

(C) IBM Corporation 1995

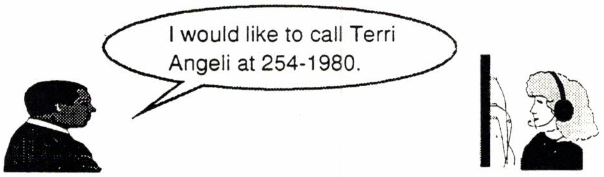
Getting Started with CPI-C

- Defining conversation characteristics - Initialize
- Begining a conversation on the client - Allocate
- Beginning a conversation on the server - Accept
- Sending data - Send
- Receiving data and status - Receive
- Ending a conversation - Deallocate

- Others
  - Forcing data to be sent - Flush
  - Exchanging confirmation - Confirm and Confirmed
  - Reporting application errors - Send_Error

Initializing a Conversation - CMINIT

- Provides information required to set up a conversation:
  - Who's my partner LU?
  - What's the TP name on the partner machine?
  - What network characteristics do I want?
- Uses CPI-C side information
- Allows pre-defined values to be changed with **SET** calls.


I would like to call Terri Angeli at 254-1980.

CMINIT Details

Call	State(s) issued from	Next state if successful
cminit	Reset	Initialize

```
cminit(                     /* Initialize_Conversation*/
      conversation_ID, /* O: conversation ID       */
      sym_dest_name,  /* I: sym_dest_name        */
      return_code);        /* O: return code            */
```

(C) IBM Corporation 1995

Client Conversation Setup

CPI-C Side Information

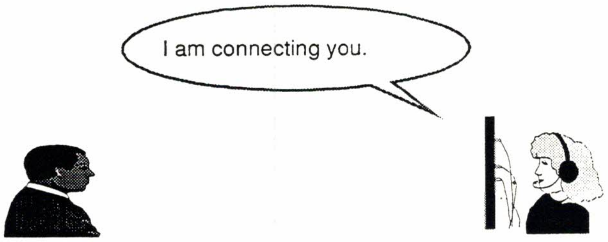
Symbolic Destination Name	**Partner LU Name**	**Mode Name**	**TP Name**
JOHNQ	*USIBMNR.NR43TRU*	*#INTER*	*ATELLD*
ANAMED	*USIBMNR.NR31EC7I*	*#INTER*	*ANAMED*
FILE_SERVER	*USIBMNR.NR53EDDI*	*#BATCH*	*AFTPD*

★ Automatic way to provide the three pieces of information required for a conversation!

(C) IBM Corporation 1995

Allocating a Conversation - CMALLC

- Assigns a session for the conversation
  - no application data flows
- Conversation appears active to the client


I am connecting you.

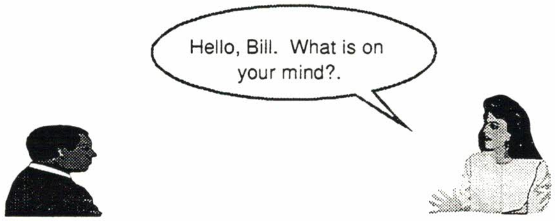
CMALLC Details

Call	State(s) issued from	Next state if successful
cmallc	Initialize	Send

```
cmallc(                    /*Allocate_Conversation*/
     conversation_ID,  /* I: conversation ID        */
     return_code);       /* O: return code             */
```

Accepting a Conversation - CMACCP

- Answers conversation request from partner
- Initializes conversation characteristics for the server side
- Conversation appears active to the server


Hello, Bill. What is on your mind?.

CMACCP Details

Call	State(s) issued from	Next state if successful
cmaccp	Reset	Receive

```
cmaccp(                     /* Accept_Conversation */
     conversation_ID,  /* I: conversation ID       */
     return_code);        /* O: return code            */
```

Sending Data - CMSEND

- Sends data to an APPC buffer
  - APPC buffers are transparent to both sending and receiving programs
- Accumulates data until
  - There is enough data to transmit to the partner
  - The sending program moves out of send state

Terri, I have the sales figures you requested.

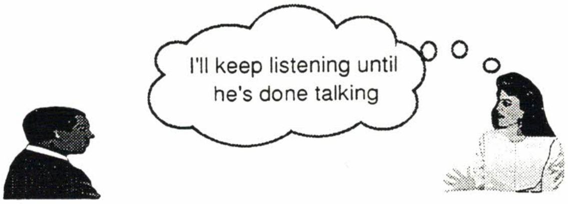
CMSEND Details

Call	State(s) issued from	Next state if successful
cmsend	Send	Send

```
cmsend(                  /* Send_Data             */
     conversation_ID,  /* I: conversation ID    */
     buffer,           /* I: data to send       */
     send_length,      /* I: size of data       */
     control_info_rcvd, /* O: control information */
     return_code);     /* O: return code        */
```

Receiving Data and Status - CMRCV

- Waits until your buffer is filled or a complete record arrives
- Returns four pieces of information
  - Return code
  - Data Received
  - Data in buffer
  - Status received


I'll keep listening until he's done talking

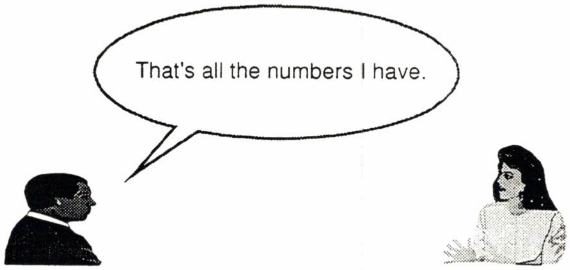
CMRCV Details

Call	State(s) issued from	Next state if successful
cmrcv	Send or Receive	Receive, Confirm, Send, or Reset

```
cmrcv(                    /* Receive                */
     conversation_ID,  /* I: conversation ID      */
     buffer,              /* O: data buffer         */
     requested_length, /* I: data buffer length   */
     data_received,     /* O: was datareceived?*/
     received_length,  /* O: size of data rcvd    */
     status_received,   /* O: was status rcvd?    */
     control_info_rcvd, /* O: control information */
     return_code);       /* O: return code          */
```

Flushing Data to the Partner - CMFLUS

- Forces data to be sent
  - Allows the other side to start processing the data
- Not normally used


That's all the numbers I have.

(C) IBM Corporation 1995

CMFLUS Details

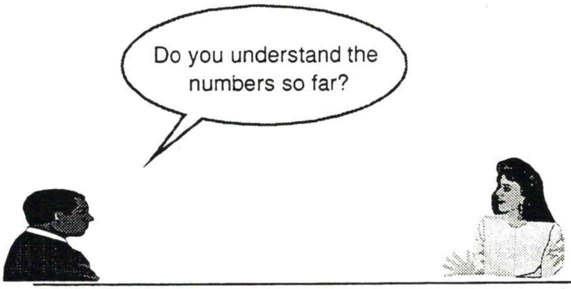
Call	State(s) issued from	Next state if successful
cmflus	Send	Send

```
cmflus(                  /* Flush_Data            */
     conversation_ID,  /* I: conversation ID    */
     return_code);      /* O: return code        */
```

(C) IBM Corporation 1995

Requesting Confirmation - CMCFM

- Causes sender to temporarily stop sending and wait for acknowledgement from partner
- Forces data to be sent
- Does not return until partner responds


Do you understand the numbers so far?

(C) IBM Corporation 1995

CMCFM Details

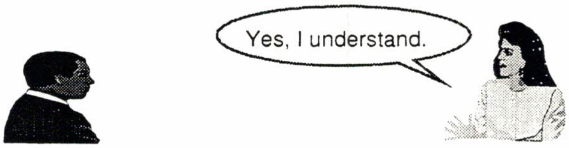
Call	State(s) issued from	Next state if successful
cmcfm	Send	Send

```
cmcfm(                    /* Confirm                 */
      conversation_ID,   /* I: conversation ID      */
      control_info_rcvd, /* O: control information*/
      return_code);      /* O: return code          */
```

(C) IBM Corporation 1995

Providing Confirmation - CMCFMD

- Verifies that the data was received and processed successfully


Yes, I understand.

(C) IBM Corporation 1995

CMCFMD Details

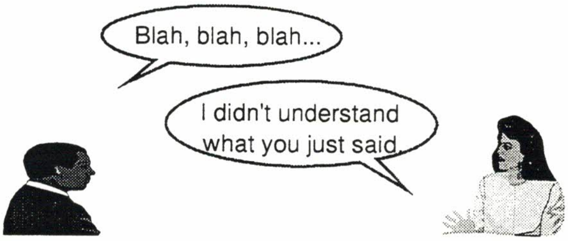
Call	State(s) issued from	Next state if successful
cmcfmd	Confirm	Receive

```
cmcfmd(                    /* Confirmed              */
     conversation_ID,  /* I: conversation ID     */
     return_code);       /* O: return code         */
```

(C) IBM Corporation 1995

Sending Error Notification - CMSERR

- Informs the partner program that a *data or logic error* was detected
  - Sends notification only, no data!
- CMSERR should be followed immediately with CMSEND


Blah, blah, blah...
I didn't understand what you just said

(C) IBM Corporation 1995

CMSERR Details

Call	State(s) issued from	Next state if successful
cmserr	Send, Receive, or Confirm	Send

```
cmserr(                  /* Send_Error              */
     conversation_ID,    /* I: conversation ID      */
     control_info_rcvd,  /* O: control information */
     return_code);       /* O: return code          */
```

★ *Only way program in receive state can take send control rather than waiting to be given it*

(C) IBM Corporation 1995

Deallocating Conversation - CMDEAL

- Ends the conversation
- Causes any remaining data to be sent

Gotta go.... Bye!

CMDEAL Details

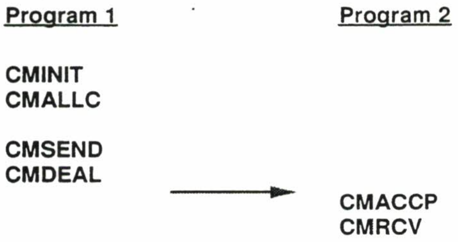
Call	State(s) issued from	Next state if successful
cmdeal	Send	Reset

```
cmdeal(                  /*Deallocate_Conversation*/
       conversation_ID,  /* I: conversation ID      */
        return_code);    /* O: return code          */
```

Classic Transaction Models

- There are five basic transaction models
  - Pipeline
  - Credit check
  - File Transfer
  - Inquiry
  - Database update
- Most distributed programs are combinations or variations of these five

Pipeline



Program 1

CMINIT
CMALLC

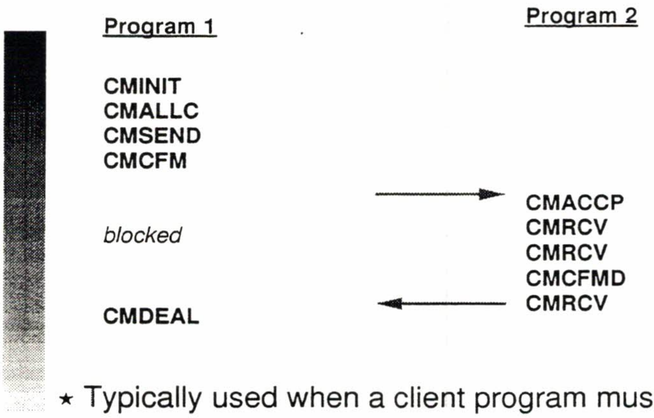
CMSEND
CMDEAL

Program 2

CMACCP
CMRCV

★ Used when no acknowledgment is needed.

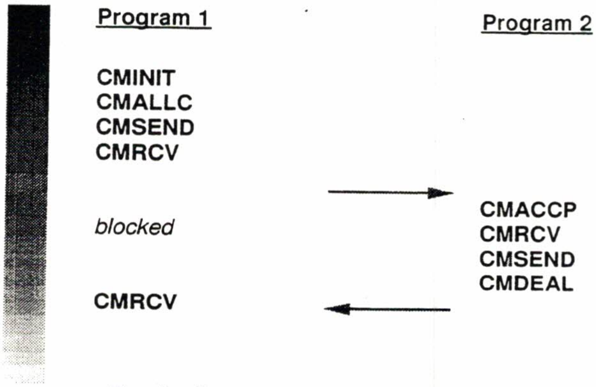
Credit Check



Program 1		Program 2
CMINIT CMALLC CMSEND CMCFM		
	→	CMACCP
blocked		CMRCV
		CMRCV
		CMCFMD
CMDEAL	←	CMRCV

- ★ Typically used when a client program must request permission to perform a specified function.

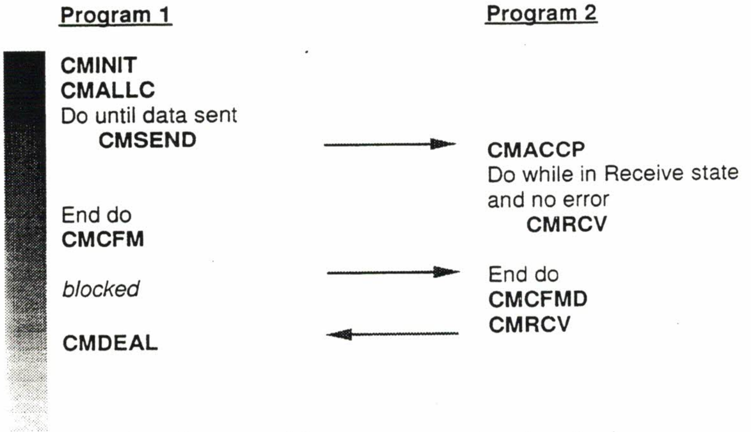
Inquiry



Program 1		Program 2
CMINIT CMALLC CMSEND CMRCV		
	→	CMACCP
blocked		CMRCV
		CMSEND
		CMDEAL
CMRCV	←	

- ★ Typically used to request information from a server program

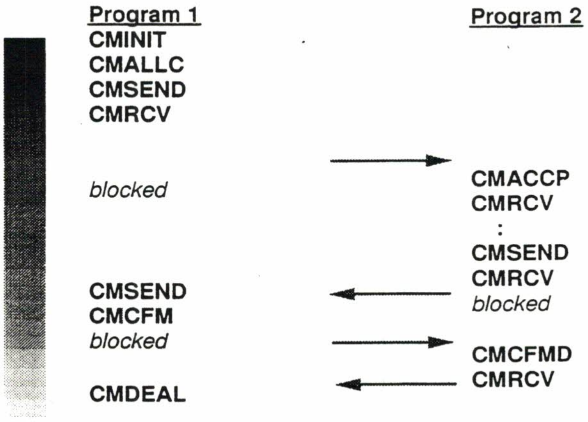
File Transfer



Program 1		Program 2
CMINIT		
CMALLC		
Do until data sent		
CMSEND	→	**CMACCP**
		Do while in Receive state and no error
End do		**CMRCV**
CMCFM		
	→	End do
blocked		**CMCFMD**
CMDEAL	←	**CMRCV**

- ⋆ Used to send large amounts of data from one location to another.

Database Update



Program 1		Program 2
CMINIT		
CMALLC		
CMSEND		
CMRCV	→	**CMACCP**
blocked		**CMRCV**
		:
		CMSEND
CMSEND	←	**CMRCV**
CMCFM		*blocked*
blocked	→	**CMCFMD**
CMDEAL	←	**CMRCV**

- ⋆ Typically used to update information managed by a partner program.

"Hello World" Sample Program

- Pipeline Transaction
- C language
- Return code processing and error handling removed for simplicity.

★ Taken from CPI-C Programming in C by John Q. Walker II and Peter J. Schwaller

(C) IBM Corporation 1995

"Hello World" Sample Program - Client

```
#include <cpic.h>          /* conversation API library   */
#include <string.h>        /* strings and memory         */
#include <stdlib.h>        /* standard library           */

/* this sym_dest_name is 8 chars long & blank padded */
#define SYM_DEST_NAME  (unsigned char*)"HELLO1S "
/* this is the string we're sending to the partner         */
#define SEND_THIS      (unsigned char*)"Hello, world"

int main(void)
{
   unsigned char   conversation_ID[CM_CID_SIZE];
   unsigned char * data_buffer = SEND_THIS;
   CM_INT32   send_length =
           (CM_INT32)strlen(SEND_THIS);
   CM_RETURN_CODE  cpic_return_code;
   CM_CONTROL_INFORMATION_RECEIVED c_i_rcvd;
```

(C) IBM Corporation 1995

"Hello World" Sample Program - Client (Page 2)

```
cminit(                  /* Initialize_Conversation     */
  conversation_ID,       /* O: returned conversation ID*/
  SYM_DEST_NAME,         /* I: symbolic dest. name      */
  &cpic_return_code);    /* O: return code from this call*/
cmallc(                  /* Allocate                    */
  conversation_ID,       /* I: conversation ID          */
  &cpic_return_code);    /* O: return code from this call*/
cmsend(                  /* Send_Data                   */
  conversation_ID,       /* I: conversation ID          */
  data_buffer,           /* I: send this buffer         */
  &send_length,          /* I: length to send           */
  &c_i_rcvd,             /* O: control information      */
  &cpic_return_code);    /* O: return code from this call*/
cmdeal(                  /* Deallocate                  */
  conversation_ID,       /* I: conversation ID          */
  &cpic_return_code);    /* O: return code from this call*/
return(EXIT_SUCCESS); }
```

(C) IBM Corporation 1995

"Hello World" Sample Program - Server

```
#include <cpic.h>          /* conversation API library  */
#include <stdio.h>         /* file I/O                  */
#include <stdlib.h>        /* standard library          */

int main(void)
{
  unsigned char  conversation_ID[CM_CID_SIZE];
  unsigned char  data_buffer[100+1];
  CM_INT32       requested_length =
              (CM_INT32)sizeof(data_buffer)-1;
  CM_INT32       received_length = 0;
  CM_RETURN_CODE cpic_return_code;
  CM_DATA_RECEIVED_TYPE data_received;
  CM_STATUS_RECEIVED   status_received;
  CM_CONTROL_INFORMATION_RECEIVED c_i_rcvd;
```

(C) IBM Corporation 1995

"Hello World" Sample Program - Server (Page 2)

```
cmaccp(                  /* Accept_Conversation         */
   conversation_ID,      /*  O: returned conversation ID */
   &cpic_return_code);  /*  O: return code from this call*/
cmrcv(                   /* Receive                     */
   conversation_ID,      /*  I: conversation ID         */
   data_buffer,          /*  I: where to put received data */
   &requested_length,    /* I: maximum length to receive*/
   &data_received,       /*  O: data complete or not?   */
   &received_length,     /*  O: length of received data  */
   &status_received,     /*  O: has status changed?      */
   &c_i_rcvd,            /*  O: control information      */
   &cpic_return_code);  /*  O: return code from this call*/
data_buffer[received_length] = '\0';     /* insert the null */
(void)printf("%s\nPress a key to end..\n", data_buffer);
(void)getchar();         /* pause for any keystroke      */
return(EXIT_SUCCESS);}
```

References

- *CPI-C Reference* (SC26-4399-07)
- *CPI-C Programming in C,* Walker and Schwaller, McGraw-Hill, ISBN 0-07911733-3
- The Best of APPC, APPN and CPI-C CD-ROM (SK2T-2013-04)

Summary

- Use CPI-C since it simplifies programming
- Begin with the starter set of functions
- Base your applications on the classic transaction programs
- Use sample programs on Best of...CD

More information about APPC and APPN

- The Best of APPC, APPN and CPI-C CD-ROM (order number SK2T-2013)
- The APPC Connection
- The APPC/APPN Technical Conference World Tour '95
  - Chicago May 22-25
  - London September 4-7
  - Sydney October 16-19

LAN 0501

LAN Server 4.0 in a Warp Environement

James Norman
Norcom Consulting Inc
Alexandria, VA

LAN SERVER 4.0 IN A WARP ENVIRONMENT

The goal of a LAN is to share devices, data, programs, and ideas. This will be the rest of the presentation. We will concentrate on LAN server in an OS/ 2 environment.

We got into the LAN business because I needed in Alexandria to share a laser printer with my wife. Meanwhile, Bob, in Atlanta, wanted to share some disk space with his wife. He also wanted to share a label printer. Benning wanted to let his secretary have access to his calendar for updates. John needed to share information about the status of projects that his people were working all. From his real estate office, Charlie wanted to share a modem and a fax machine.

The commonalty among all of these and many others is the requirement to share and that each of them already had some computer system installed. Another common element is that they are all true situations. And, interestingly enough, they all were originally written on a napkin.

In each case a sketch was made. In each case, it seemed so simple. In each case, the eyes got bigger than the stomach. And, yes, in each case the idea slipped off the priority list until someone told them how to do what they wanted to do easily and simply.

These sketches make up the core of the book we are preparing.

FEATURES

LAN Server Release 4.0 is built around object technology with many enhancements for installation and operations over Release 3. For example, in a large number of cases, the software will recognize the hardware adapter correctly and for those that it won't recognize the setup procedures are easily constructed during the installation.

There is a facility to browse beyond your local domain to use resources of attached systems. And a single network logon is provided to connect with other systems, or elsewhere, without having to additionally logon. Installation and execution of public applications has been made significantly easier.

Documentation, both on-line and on paper, has been improved significantly. There are a number of transport improvements including multi-protocol support and support for TCP/IP and for support of Netbios over TCP/IP. For improved performance, there is support for the synchronous multiprocessors (SMP) as well as enhanced support for the new chips.

PLANNING

We discovered the value of planning when we received the Advanced Server for Workgroups package from IBM. In the box was a small, short book that gave the step by step instructions to install a server, and a system from Lotus called Notes.

We built a sequence of events that started with a sentence saying "what do you want to do?" Put that sentence away, you will need it later!

Second, take the drawing prepared earlier and formalize a little bit. List all of the names, and add in any physical constraints. Keep in mind that star-wired topology (independent of protocol) is your best long term solution. All these notes together make the core of your personal documentation.

DOCUMENTATION

The administrator should have the goals of the network. He should also have all hardware and software installation manuals. In many cases, particularly for Software distribution, he will need the application manuals also. There must also be a testing procedure documented for use by all involved in the installation.

The user should be familiar with the goals of the network and should have network user guide available. The user should also have guidance about the best way to report problems to the administrator.

THE FIRST SERVER

Setting up the initial server is very easy. Choose whether you want to use the Entry or the Advanced System. We will do you some guidance here. Installation amounts to little more than inserting the CD-ROM into the drive and selecting the install program. The one tuning item you should remember is that the server cache size should be all of memory less 12MB. During the installation you should also install IBM NETBIOS to assure that Windows NT and Windows for Workgroups will function properly later.

THE FIRST CLIENT

Planning for the first client should incorporate planning for the last client also. There are several types of clients and each requires its own planning and its own understanding. There is one client for OS/2, one for Windows, but there are three kinds for DOS. In this lecture we will look only at the OS/2 Client.

one of the major advantages of land server is that it supports distribution of software from the server to the five. This is called configuration, installation, and distribution (CID).

SHARING THE PRINTER

Sharing a printer with LAN Server can be very easy, if the physical configuration is right. For instance, when my configuration, with the printer on the server, needed to share the printer, I just selected the icon and said "start sharing" and answered the security questions. On the client OS/2 machine, I just browse the network, selected the printer, and assigned it a port. At the completion of that, the native OS/2 applications had a printer. Windows applications require that the Windows control panel be invoked and the printer defined to them. All applications on the client machine, therefore, had access to the printer.

One thing important to remember is that the printer must have the same description on the client machine that it has on the server. If that is true, then applications can even select features of the printer to use such as tray choice or envelope preparation.

If, however, the printer is on a client machine, that process is significantly more difficult. If the printer does not have the same definitions, then, whether on client or server, it will be very difficult to get it to work.

SHARING A FILE

Files are managed by the access control manager. Here each file is given the approval to read, write, change attributes, and several other very specific functions by user or group. I used the phrase File here, I really am referring to a sub-directory. It is the exception. To additional functions in LAN server for which support moving information from one file across a network to another. In addition, there is support for resynchronizing files for the disconnected remote user. Using the advancedLAN server system, there can be controls on the amount of disk space allocated to each user.

MODEMS

Serial devices, such as modems, can be shared across the network. However, the application must be able to recognize that the device is a LAN-attached device. The administrative technique for sharing the serial device is just like other devices. You drag an icon and drop it on a user icon, and set the access control. Simply assigning a logical port completes the task.

CD ROM

A CD ROM device requires special support because it cannot be written to, and yet has the characterics of being removable like a diskette. However, LAN Server is so good that the authorized user simply browses until the device is found, and assigns a drive. This is as easy to use as any network operating system on the market today.

NOTES

One of the major future functions of a local area network is the sharing of ideas. The premier product today which supports this is the notes product from Lotus Development Corp.. Notes was designed to support group ware and the sharing of ideas. LAN server is a good system to support sharing, in general, so it is an ideal base for notes.

Lotus Development Corporation's Notes product has been packaged for the LAN server. The defaults and run time are set for LAN server. In fact, it is actually fairly well tuned in an out-of-box condition. This is one case where the products from different vendors work very well together. If Lotus Notes is on your agenda, get the prepackaged Advanced Server for Workgroups which has a pre-setup install procedure which works well and fast. It gives you all the benefits of an OS/2 system plus a LAN Server system and a Notes system. They generally run about 30 to 60 days after shipment of a release before the integration is completed. When it is complete, it works and works well saving you time and money.

SHARING APPLICATIONS

Sharing an application across a local area network is not much more difficult than installing the application, telling the server about it, and granting access to the group of users. what more could you ask of a system?

THE TRAVELING LAN

When you travel, you can use a product called LAN distance to connect your laptop to your local area network. LAN distance allows applications to be run transparent. Connections that can be used include asynchronous, synchronous, x.25, and I S D N. Because you are connected remotely, the security support is extensive. It even includes call back functions. Because this software is so well done, it also supports an ad hoc LAN to LAN connection.

SECURITY

In LAN Server, all resources are considered off limits until they are explicitly granted permission to be used. This is different from some other network operating systems. All access provisions that are granted are granted specifically to the resource. When any one is connected remotely, there is substantial verification of the authenticity of the user.

PERFORMANCE

In at least one of the recent benchmarks, LAN server performed functions faster than other products in the same class. Part of this speed comes from the fact that the majority of the software is 32 bits already. Additional speed comes from the management of cache.

There are many techniques available to improve speed in an installed installations which would not show in a benchmark. For instance, if the server had two adapter cards in it, the cards would be able to perform load balancing if operating on Netbios. The disk access mechanism is distributed to avoid burdening one disk arm.

More important than this is the fact that there is a tuning assistant which will help the administrator maximize the performance of his server. In addition there are a number of white papers which specifically address the subject of performance. They can be acquired from:

TRENDS AND DIRECTIONS

As LAN server matures, there will be more application sharing. There will be more ease of use functions made available. There will be more performances enhancements to take advantage of higher speed processors, files and other I/O as well as SMP. In addition there will be further refinements to the install process making it even get easier and more manageable.

SUMMARY

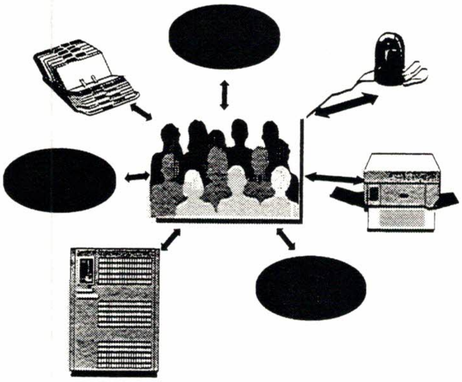
In summary, LAN server for brings industrial strength performance, simplified administration, an opportunity for flexible growth, as well as open interfaces to support other systems such as Windows for Workgroups. Today LAN server supports a wide array of hardware components and will be able to support more in the future. It has built in peer to peer support for Workgroups. This will be enhanced further as it gets closer to OS/2.

Not only that, but it is easy to install, use, maintain, and operate!

LAN Server Release 4.0 from IBM, it is a technological winner. You can help make it a marketplace winner.

Sketch

- Jim wanted to share a printer
- Bob wanted to share files
- Benning wanted to share his calendar
- John wanted to keep up to date on the projects ...
- Charlie wanted to share a modem and a fax machine
-
-



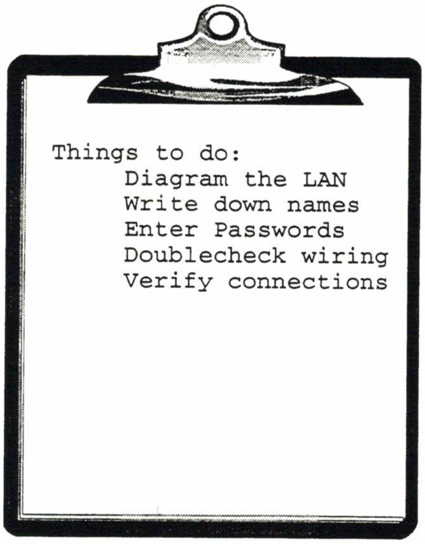
Most LANs start on a napkin!

Features

- Object Technology
- Installation Assists
  - LAN Adapter Detection
  - More Adapters Supported
- Operations Assists
  - Multi-Domain Browse
  - Single Network Logon
  - Public Applications
- Documentation
- Transport Improvements
  - Multi-Protocol
  - TCP/IP
  - Netbios for TCP/IP
  - Netbios caching
- Performance
  - SMP Support
  - Pentium Exploitation
  - Fast Clients

Planning

- Write down goals
  - File them for later
- Draw the LAN
- List all names
- Examine Physical Facility
  - Star-wired cabling
- Check Goals



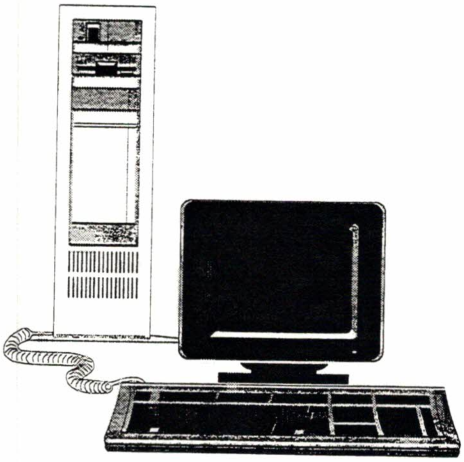
Things to do:
Diagram the LAN
Write down names
Enter Passwords
Doublecheck wiring
Verify connections

Documentation

- Administrator
  - Goals of Network
  - Installation Manuals
    - Hardware
    - Software
    - Testing Procedures
  - Operation Manuals
    - Trouble Reports
    - Trouble Database
  - Application Pubs
- User
  - Network User Guide
    - Paper and
    - Electronic
  - Application Manuals
    - Purchased
    - In-house Written
  - Trouble Reports

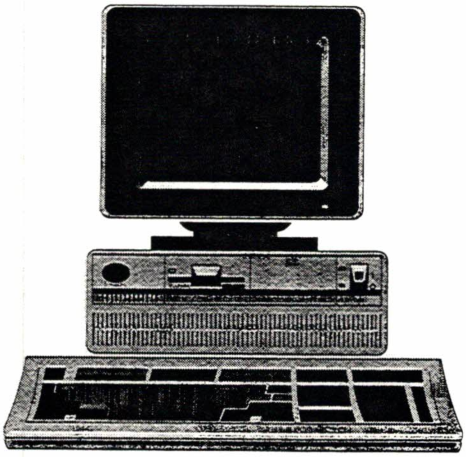
Server

- Entry and Advanced Systems
  - Administration
    - Devices
    - Users
  - Data Administration
    - Shared Programs
  - Password Control



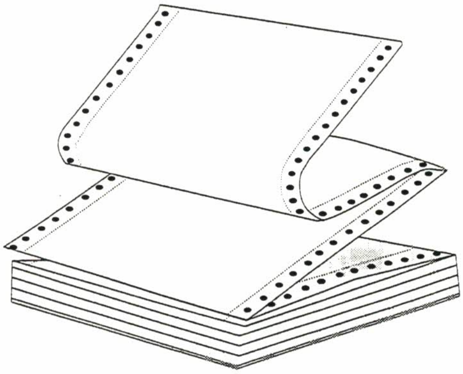
Client

- New clients for DOS, Windows
    - Performance, functions
- Windows for Workgroups and NT clients support
- 32 bit development environment



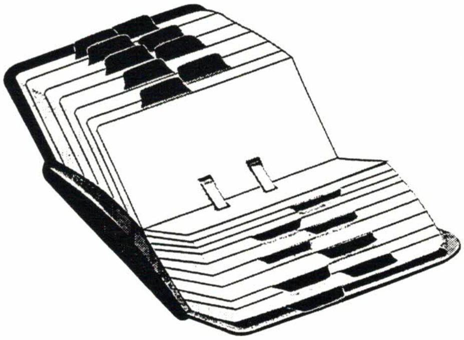
Printing

- Drag a template, setup a printer
- Match the values!



Files

- Access Control Manager
- Movestuf
- Remote file synchronization
- Disk Space Limits
  - Advanced



Modems

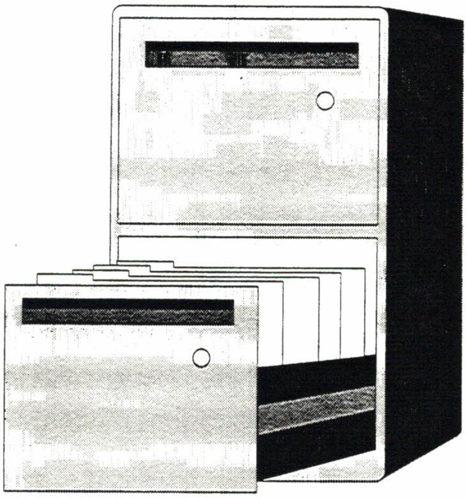
- Serial Device Support
    - Applications must recognize function
    - Examples: Faxworks, Lotus Desktop
  - Drag and Drop,
  - Set Access Control
  - Assign a logical port

CD ROM

- Fully supported as a read-only device
- Set access control
- User browses and
- assigns a drive
- Easy to use

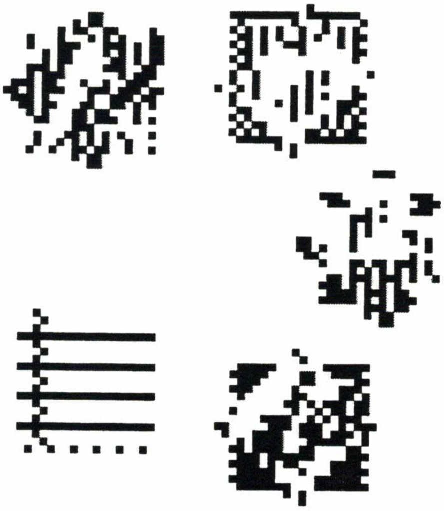
Notes

- Packaged for LAN Server 4.0
- Set defaults and run
- Tune after installation



Sharing Applications

- Install the Application
- Tell LAN Server about it
- Grant access to it by dragging the user to the icon



IBM
OS/2
OS/2 Warp

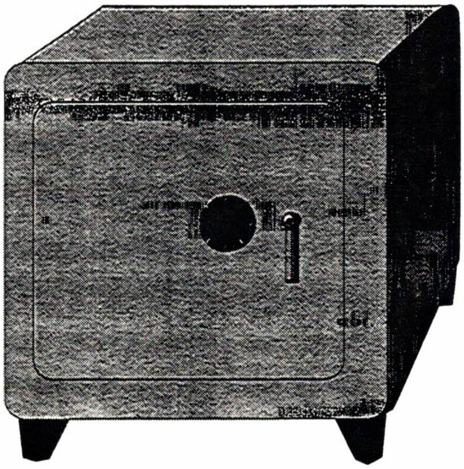
LAN Distance

- Applications run transparently
- Connections via
  - Asynchronous
  - Synchronous
  - X.25 and ISDN
- Extensive Security
- Supports ad hoc LAN over WAN

Norcom Consulting Inc

Security

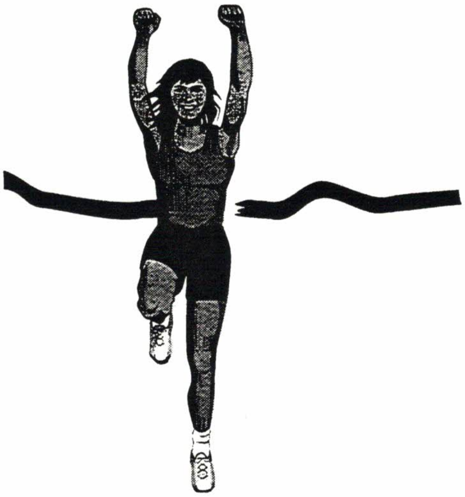
- All resources are "off limits" until granted
- All grants are specific to the resource
- Extensive Remote checking



Norcom Consulting Inc

Performance

- 32 bit Programs
- Caching on Client
- Routable Protocol
- Adapter Load Balancing
- Distributed Disk Accesses
- Tuning Assistant



Trends & Directions

- More application sharing
- SMP Performance enhancements
- Packaging improvements
- Installation Assistants

Summary

- Industrial-strength performance
- Simplified Administration
- Flexible Growth
- Open Interfaces to support other systems
- Extensive hardware supported
- Built-in peer to peer for workgroups

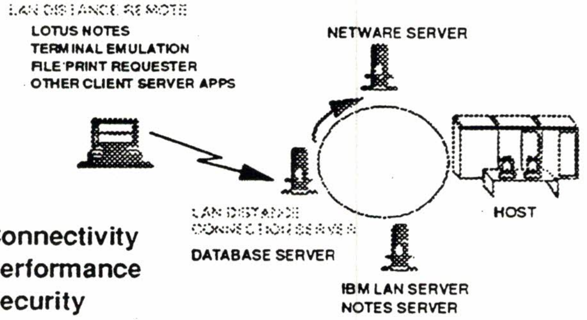
And it is easy to install, use, maintain and operate!

LAN 0508

LAN Distance Overview

Bob Spickelmier
IBM

LAN Distance Overview


LAN DISTANCE REMOTE
LOTUS NOTES
TERMINAL EMULATION
FILE PRINT REQUESTER
OTHER CLIENT SERVER APPS
NETWARE SERVER
LAN DISTANCE
CONNECTION SERVER
HOST
DATABASE SERVER
IBM LAN SERVER
NOTES SERVER

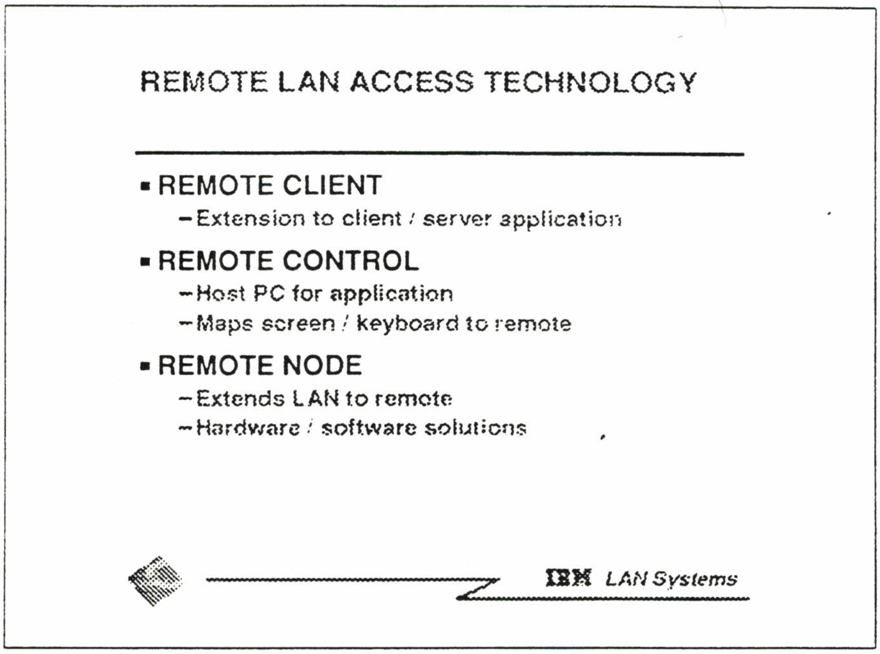
Connectivity
Performance
Security
Features / Ease of Use
Cost Per Connection

IBM *LAN Systems*

AGENDA

- REMOTE LAN ACCESS TECHNOLOGY
- LAN DISTANCE FEATURES
- FUTURE DIRECTIONS
- IMPLEMENTATION EXAMPLES
- SUMMARY

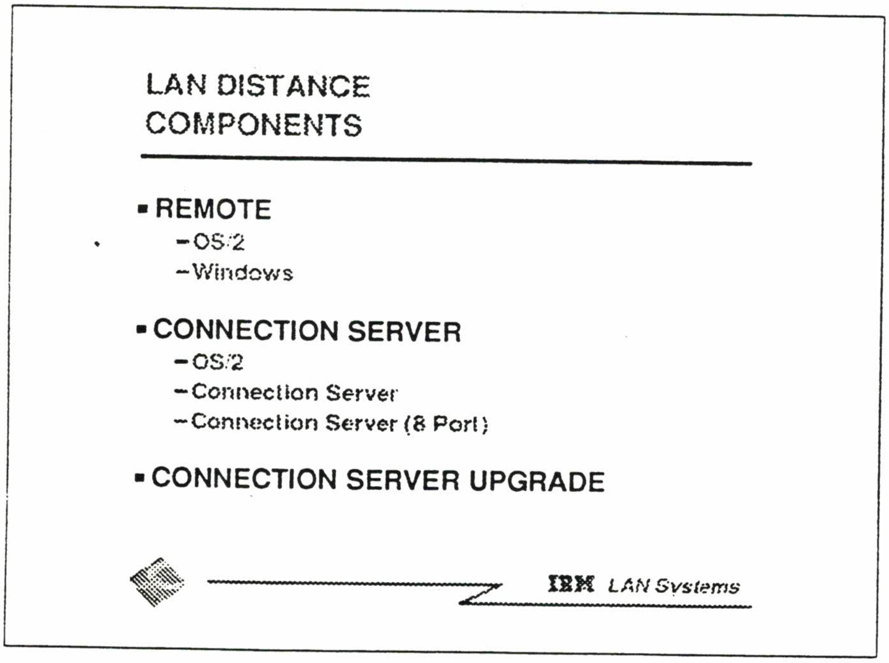
IBM *LAN Systems*



REMOTE LAN ACCESS TECHNOLOGY

- REMOTE CLIENT
  - Extension to client / server application
- REMOTE CONTROL
  - Host PC for application
  - Maps screen / keyboard to remote
- REMOTE NODE
  - Extends LAN to remote
  - Hardware / software solutions

IBM *LAN Systems*

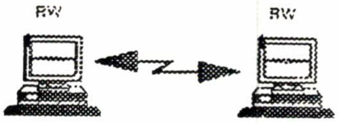


LAN DISTANCE COMPONENTS

- REMOTE
  - OS/2
  - Windows
- CONNECTION SERVER
  - OS/2
  - Connection Server
  - Connection Server (8 Port)
- CONNECTION SERVER UPGRADE

IBM *LAN Systems*

LAN DISTANCE REMOTE

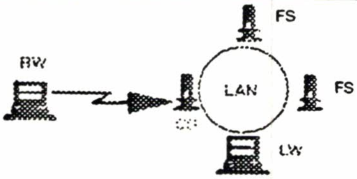


OS/2 2.X AND
MS WINDOWS

- Enables a workstation not physically connected to a LAN to dial-in and operate as if it were on a LAN
- Enables creation of a virtual LAN for point-to-point connectivity
- Two concurrent connections supported for creation of virtual networks (OS/2)

IBM LAN Systems

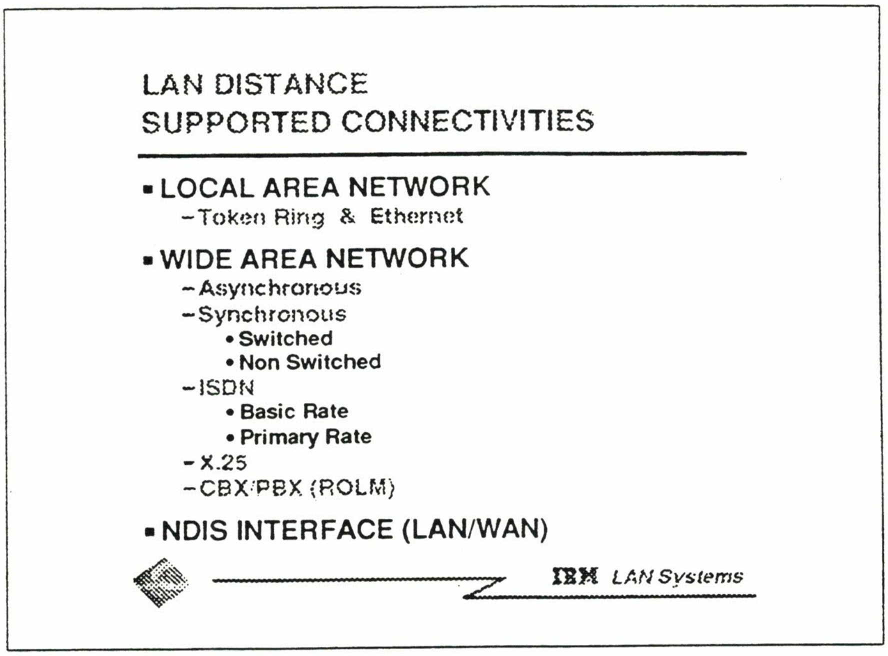
LAN DISTANCE CONNECTION SERVER



OS/2 2.X

- ESTABLISHES A CONNECTION AND ROUTES TRAFFIC BETWEEN THE LAN AND WAN
- ENFORCES SECURITY ON/OFF THE LAN
- MANAGES THE "OUTER" NETWORK
- PROVIDES AUDIT TRAILS AND ADMINISTRATION
- LAN WORKSTATION TO DIAL OFF LAN
- TWO VERSIONS
  - Connection Server
  - Connection Server (8 Port)

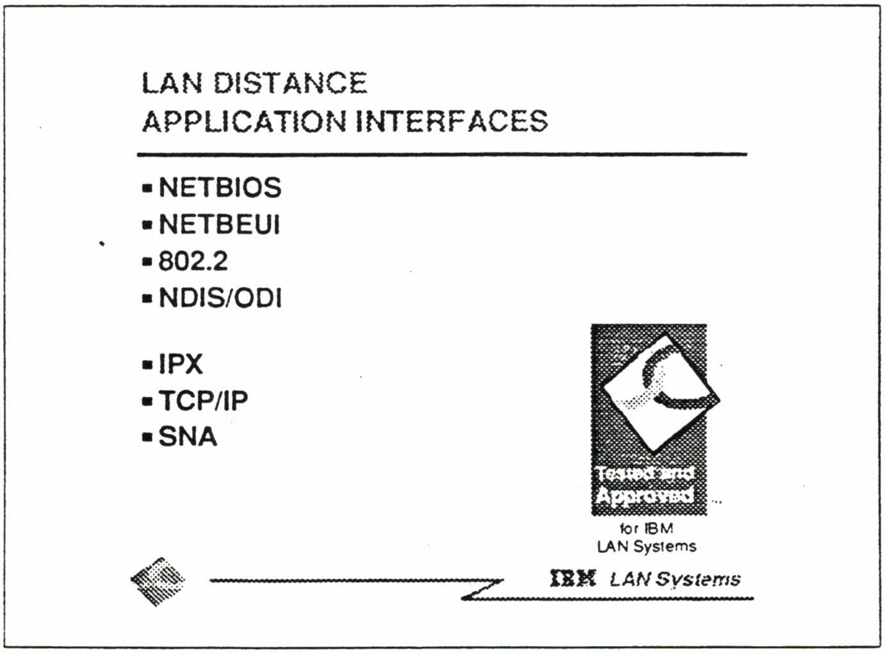
IBM LAN Systems



LAN DISTANCE
SUPPORTED CONNECTIVITIES

- LOCAL AREA NETWORK
  - Token Ring & Ethernet
- WIDE AREA NETWORK
  - Asynchronous
  - Synchronous
    - Switched
    - Non Switched
  - ISDN
    - Basic Rate
    - Primary Rate
  - X.25
  - CBX/PBX (ROLM)
- NDIS INTERFACE (LAN/WAN)

IBM *LAN Systems*



LAN DISTANCE
APPLICATION INTERFACES

- NETBIOS
- NETBEUI
- 802.2
- NDIS/ODI

- IPX
- TCP/IP
- SNA

Tested and Approved for IBM LAN Systems

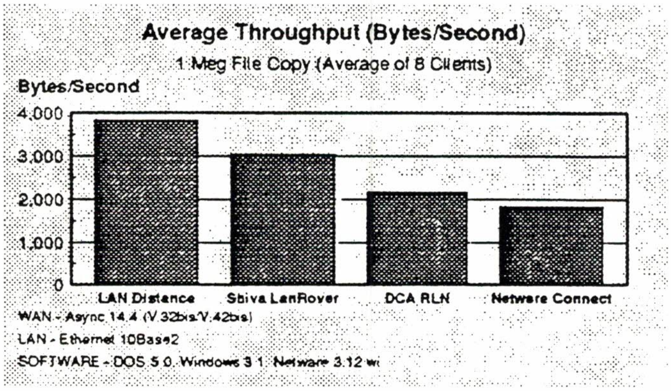
IBM *LAN Systems*

LAN DISTANCE
SECURITY OPTIONS

- USER ID/PASSWORD
- CALL BACK
- PASSWORD ENCRYPTION
- WORKSTATION ADDRESS IDENTIFICATION
- VALID LOGON TIME INTERVALS
- UNAUTHORIZED LOGON LIMIT
- ACCESS PRIVILEGE LEVELS
- CONFIGURABLE LOGON PARAMETERS
- SESSION AUTHENTICATION
- SECURITY APPLICATION EXIT
- LAN APPLICATION SECURITY

IBM *LAN Systems*

LAN DISTANCE
PERFORMANCE COMPARISON

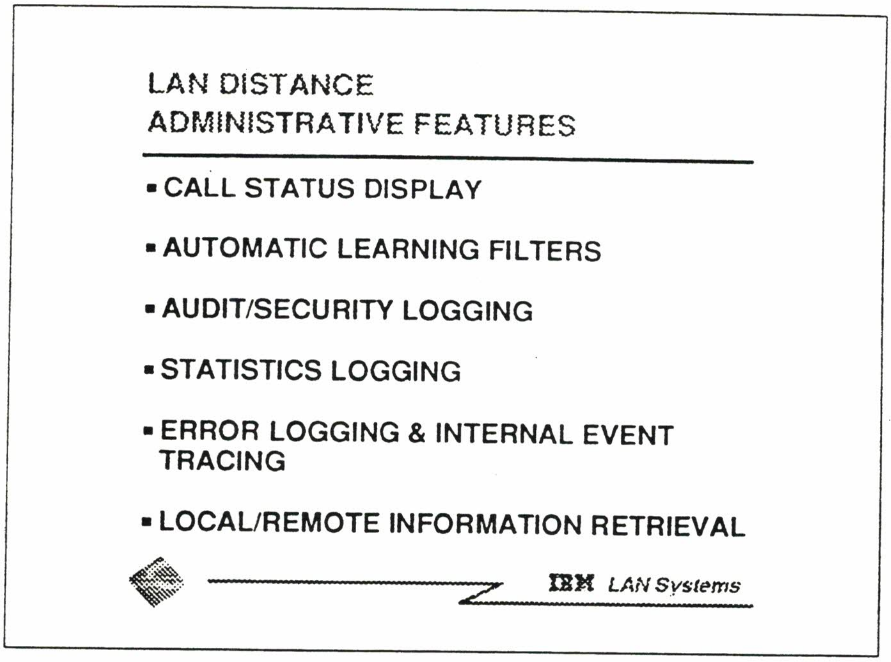


WAN - Async 14.4 (V.32bis/V.42bis)

LAN - Ethernet 10Base2

SOFTWARE - DOS 5.0, Windows 3.1, Netware 3.12 wi

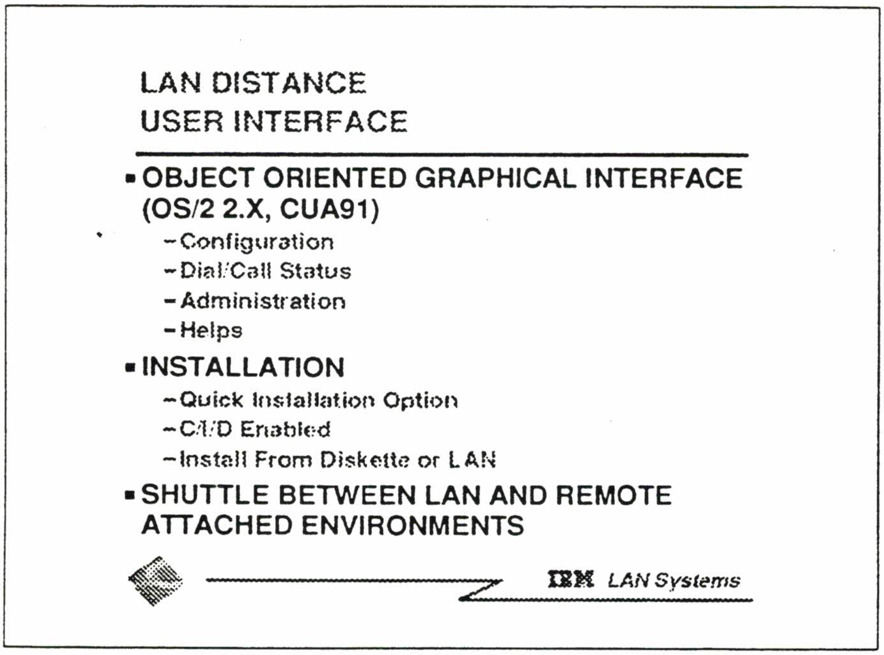
IBM *LAN Systems*



LAN DISTANCE
ADMINISTRATIVE FEATURES

- CALL STATUS DISPLAY
- AUTOMATIC LEARNING FILTERS
- AUDIT/SECURITY LOGGING
- STATISTICS LOGGING
- ERROR LOGGING & INTERNAL EVENT TRACING
- LOCAL/REMOTE INFORMATION RETRIEVAL

IBM LAN Systems

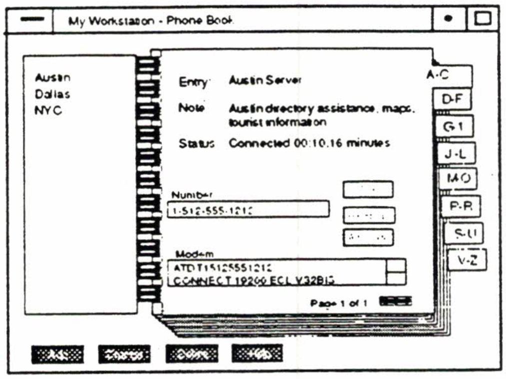


LAN DISTANCE
USER INTERFACE

- OBJECT ORIENTED GRAPHICAL INTERFACE (OS/2 2.X, CUA91)
  - Configuration
  - Dial/Call Status
  - Administration
  - Helps
- INSTALLATION
  - Quick Installation Option
  - C/I/D Enabled
  - Install From Diskette or LAN
- SHUTTLE BETWEEN LAN AND REMOTE ATTACHED ENVIRONMENTS

IBM LAN Systems

LAN DISTANCE
Phone Book and Call Status Screen


My Workstation - Phone Book
Austin
Dallas
NYC
A-C
D-F
G-I
J-L
M-O
P-R
S-U
V-Z
Page 1 of 1

Entry: Austin Server

Note: Austin directory assistance, maps, tourist information

Status: Connected 00:10.16 minutes

Number: 1-512-555-1212

Modem:
ATDT15125551212
CONNECT 19200 ECL V32BIS

IBM LAN Systems

LAN DISTANCE
FUTURE DIRECTIONS

- **FEATURES & EASE OF USE**
  - Shared sucurity database
  - Inactivity timer
  - Configuration checking utility
  - Modem support & line speed
- **EXPANDED CLIENT SUPPORT**
- **EXPANDED ADAPTER SUPPORT**
  - LAN adapter support
  - Connectivity

IBM LAN Systems

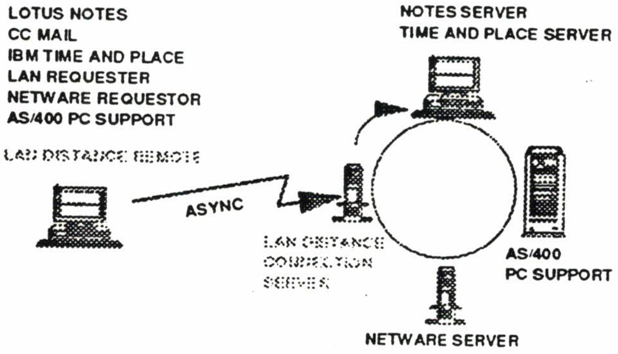
LAN DISTANCE

IMPLEMENTATION EXAMPLES

IBM LAN Systems

TRAVELING PROFESSIONAL

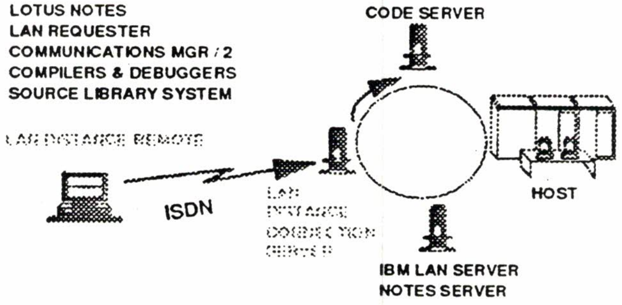
LOTUS NOTES
CC MAIL
IBM TIME AND PLACE
LAN REQUESTER
NETWARE REQUESTOR
AS/400 PC SUPPORT


NOTES SERVER
TIME AND PLACE SERVER
LAN DISTANCE REMOTE
ASYNC
LAN DISTANCE CONNECTION SERVER
AS/400 PC SUPPORT
NETWARE SERVER

IBM LAN Systems

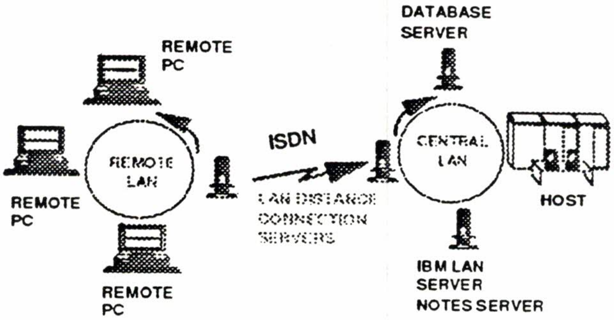
PROGRAMMER - WORK AT HOME

LOTUS NOTES
LAN REQUESTER
COMMUNICATIONS MGR / 2
COMPILERS & DEBUGGERS
SOURCE LIBRARY SYSTEM

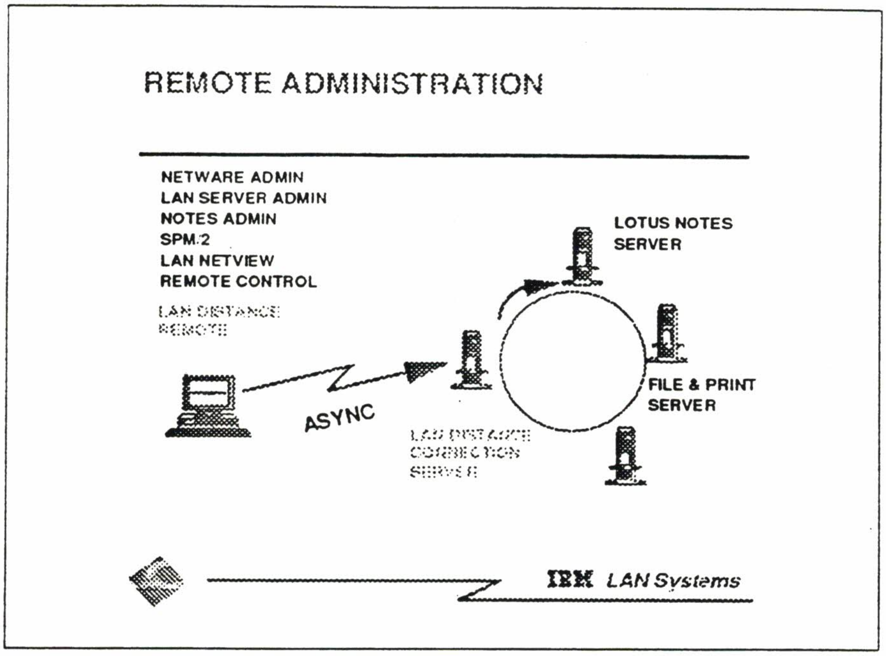

CODE SERVER
LAN DISTANCE REMOTE
ISDN
LAN DISTANCE CONNECTION SERVER
HOST
IBM LAN SERVER
NOTES SERVER

IBM *LAN Systems*

SMALL BRANCH OFFICE


DATABASE SERVER
REMOTE PC
REMOTE LAN
REMOTE PC
REMOTE PC
ISDN
LAN DISTANCE CONNECTION SERVERS
CENTRAL LAN
HOST
IBM LAN SERVER
NOTES SERVER

IBM *LAN Systems*



REMOTE ADMINISTRATION

NETWARE ADMIN
LAN SERVER ADMIN
NOTES ADMIN
SPM/2
LAN NETVIEW
REMOTE CONTROL

LAN DISTANCE
REMOTE

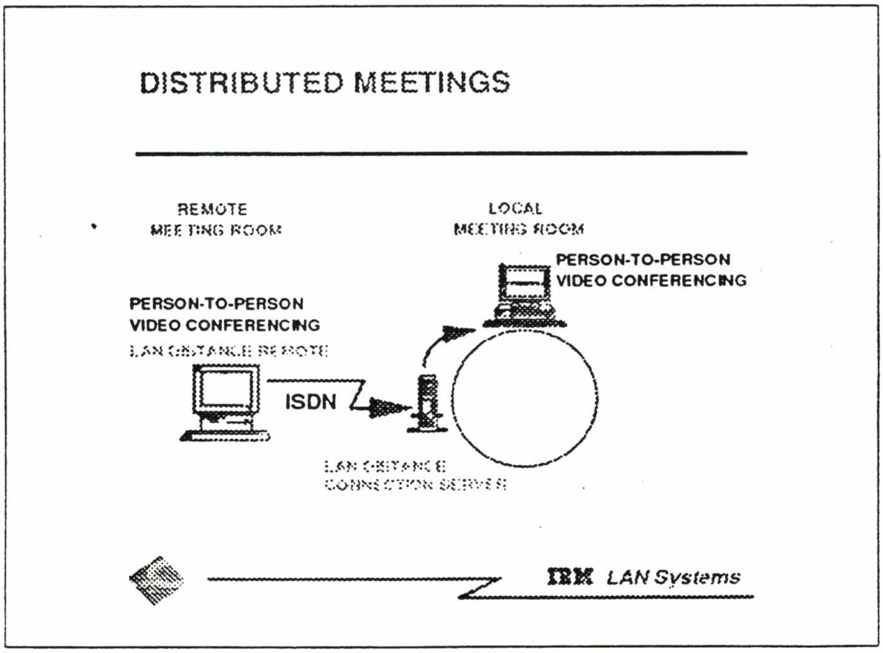
ASYNC

LAN DISTANCE
CONNECTION
SERVER

LOTUS NOTES
SERVER

FILE & PRINT
SERVER

IBM *LAN Systems*



DISTRIBUTED MEETINGS

REMOTE
MEETING ROOM

LOCAL
MEETING ROOM

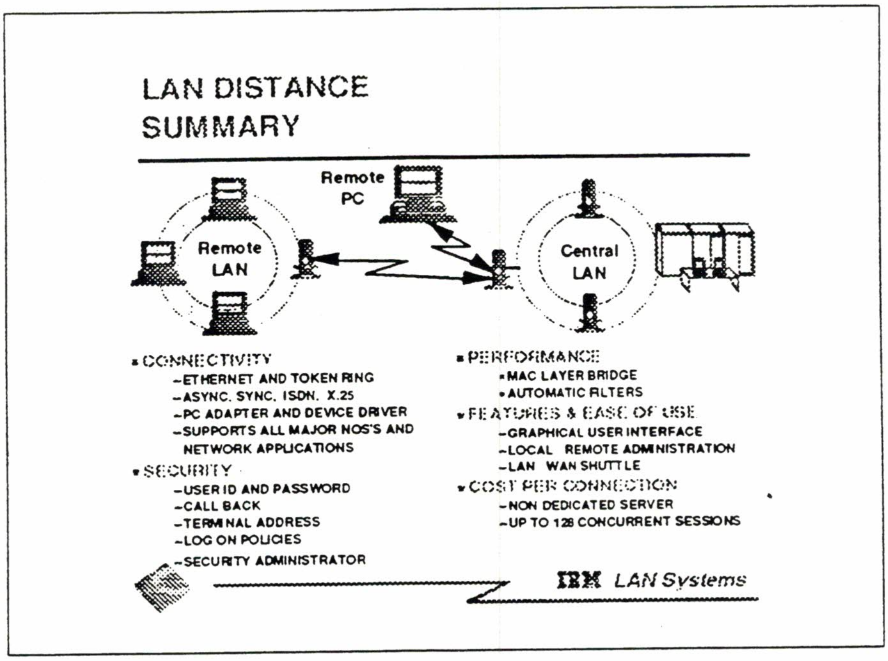
PERSON-TO-PERSON
VIDEO CONFERENCING

PERSON-TO-PERSON
VIDEO CONFERENCING
LAN DISTANCE REMOTE

ISDN

LAN DISTANCE
CONNECTION SERVER

IBM *LAN Systems*



LAN DISTANCE SUMMARY

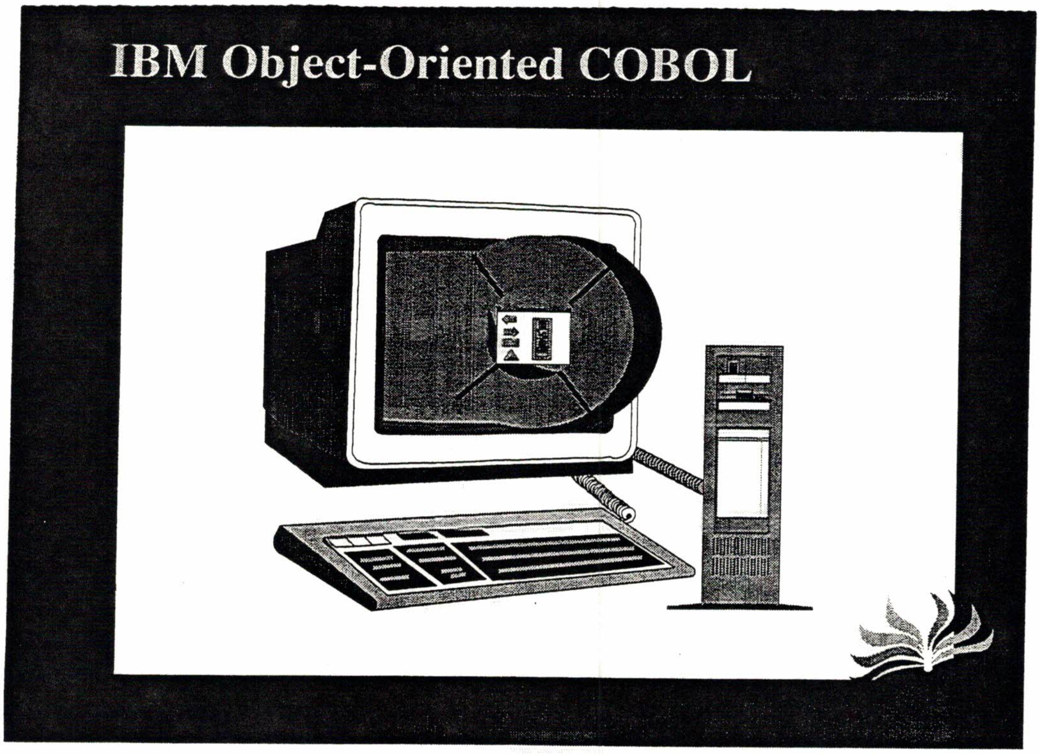
- CONNECTIVITY
  - ETHERNET AND TOKEN RING
  - ASYNC. SYNC. ISDN. X.25
  - PC ADAPTER AND DEVICE DRIVER
  - SUPPORTS ALL MAJOR NOS'S AND NETWORK APPLICATIONS
- SECURITY
  - USER ID AND PASSWORD
  - CALL BACK
  - TERMINAL ADDRESS
  - LOG ON POLICIES
  - SECURITY ADMINISTRATOR
- PERFORMANCE
  - MAC LAYER BRIDGE
  - AUTOMATIC FILTERS
- FEATURES & EASE OF USE
  - GRAPHICAL USER INTERFACE
  - LOCAL REMOTE ADMINISTRATION
  - LAN WAN SHUTTLE
- COST PER CONNECTION
  - NON DEDICATED SERVER
  - UP TO 128 CONCURRENT SESSIONS

IBM *LAN Systems*

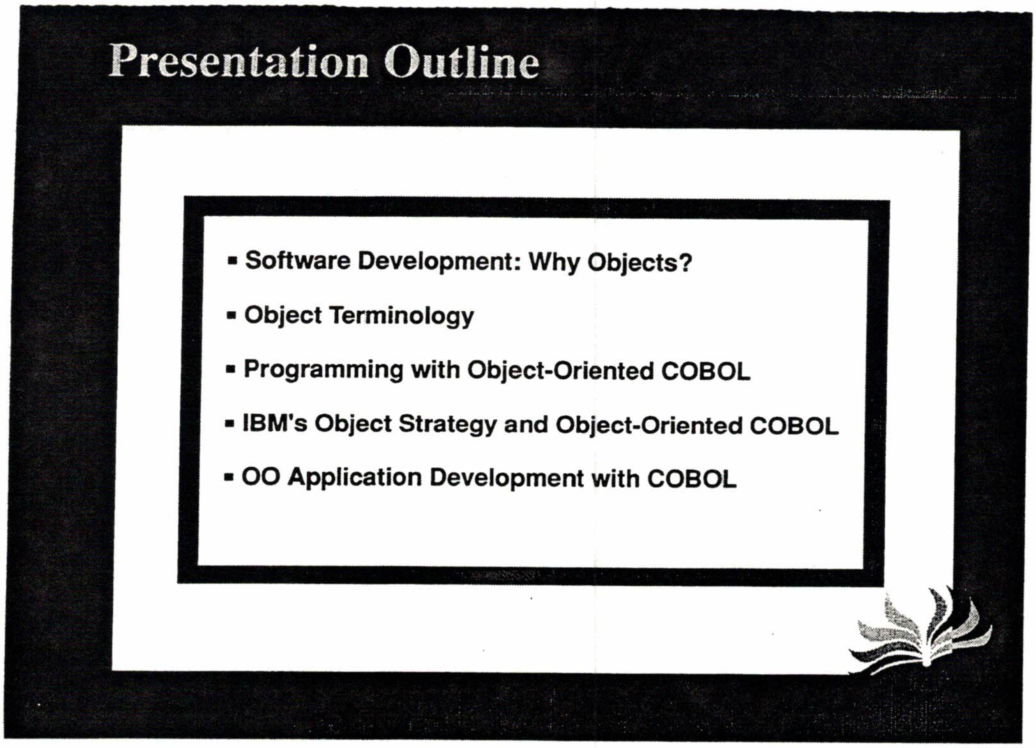
PRO 0509

IBM COBOL Goes Object-Oriented

Ernie Escuton
IBM

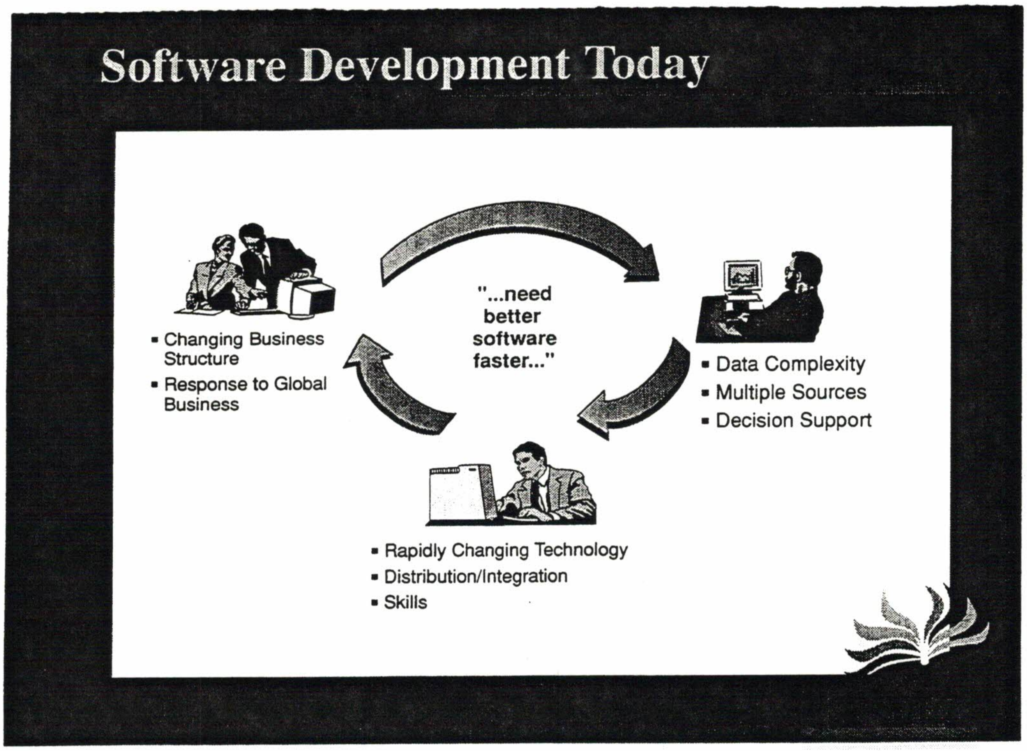


IBM Object-Oriented COBOL



Presentation Outline

- Software Development: Why Objects?
- Object Terminology
- Programming with Object-Oriented COBOL
- IBM's Object Strategy and Object-Oriented COBOL
- OO Application Development with COBOL



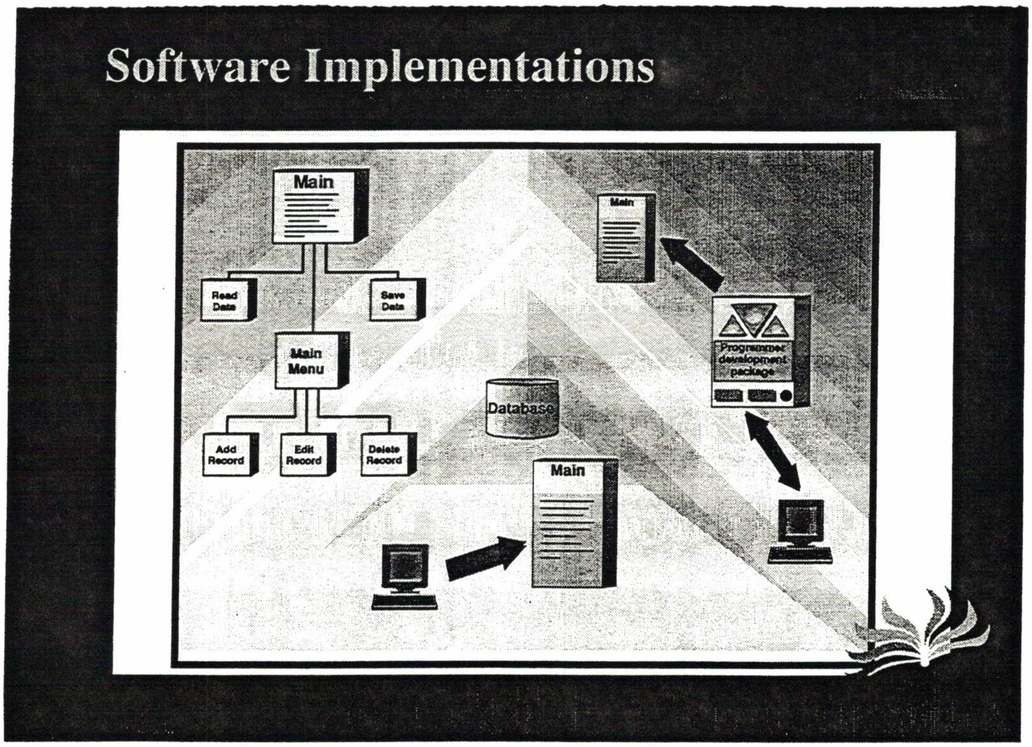
Software Development Today

"...need better software faster..."

- Changing Business Structure
- Response to Global Business

- Data Complexity
- Multiple Sources
- Decision Support

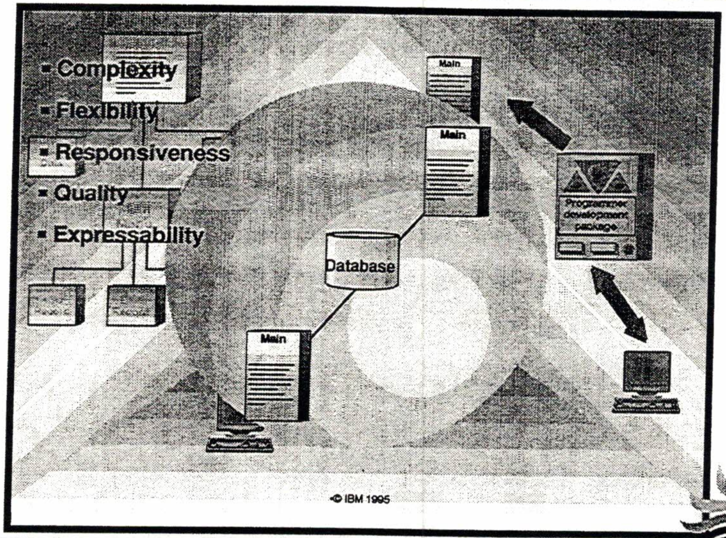
- Rapidly Changing Technology
- Distribution/Integration
- Skills



Software Implementations

Main
Read Data
Save Data
Main Menu
Add Record
Edit Record
Delete Record
Database
Main
Main
Programmer development package

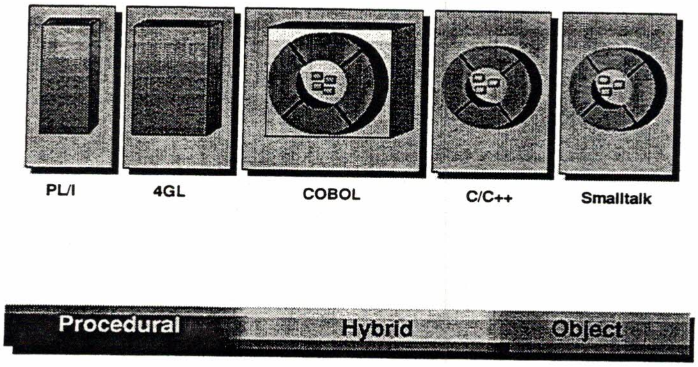
Why Objects?



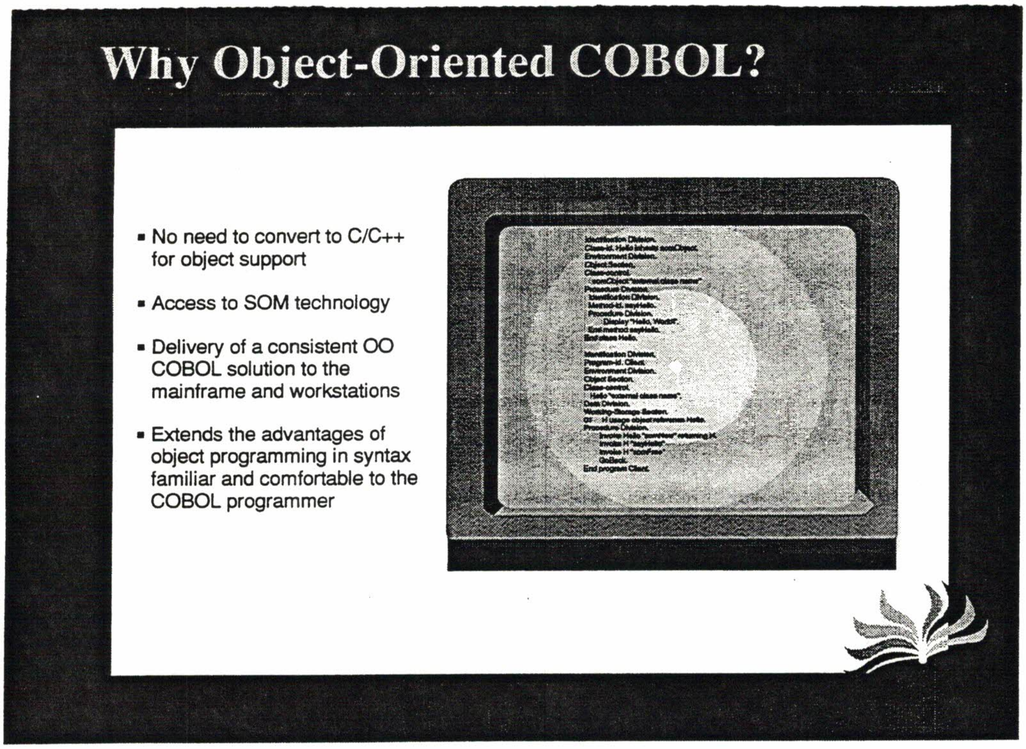
- Complexity
- Flexibility
- Responsiveness
- Quality
- Expressability

Main
Main
Main
Database
Programmer development package
© IBM 1995

Object Programming Languages



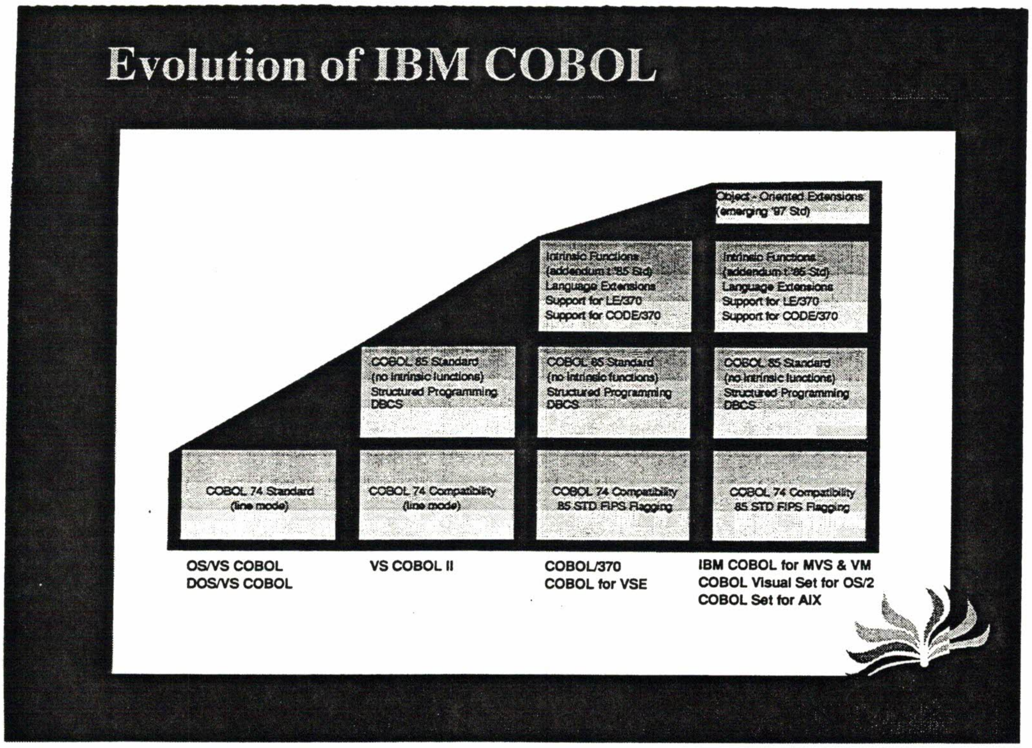
PL/I
4GL
COBOL
C/C++
Smalltalk
Procedural
Hybrid
Object



Why Object-Oriented COBOL?

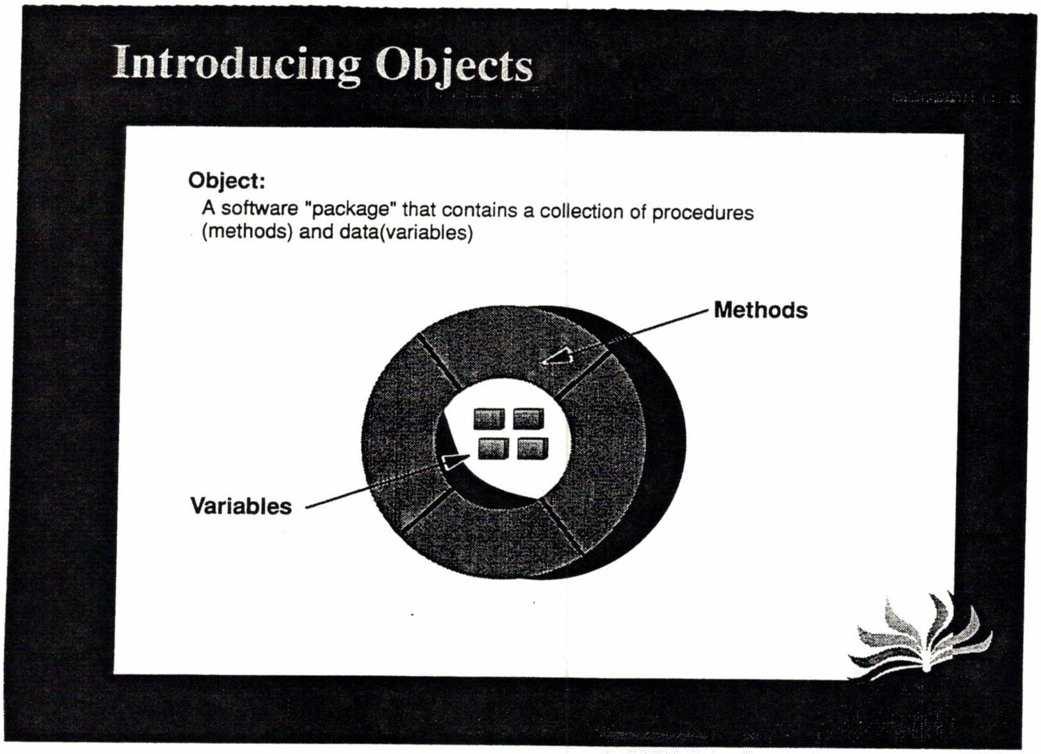
- No need to convert to C/C++ for object support
- Access to SOM technology
- Delivery of a consistent OO COBOL solution to the mainframe and workstations
- Extends the advantages of object programming in syntax familiar and comfortable to the COBOL programmer

```
Identification Division.
Class-id. Hello inherits somObject.
Environment Division.
Object Section.
Class-control.
  somObject "external class name".
Procedure Division.
  Identification Division.
  Method-id. sayHello.
  Procedure Division.
      Display "Hello, World!".
  End method sayHello.
End class Hello.

Identification Division.
Program-id. Client.
Environment Division.
Object Section.
Class-control.
  Hello "external class name".
Data Division.
Working-Storage Section.
01    H usage object reference Hello.
Procedure Division.
     Invoke Hello "somNew" returning H.
     Invoke H "sayHello".
     Invoke H "somFree".
     GoBack.
End program Client.
```



Evolution of IBM COBOL

OS/VS COBOL DOS/VS COBOL	VS COBOL II	COBOL/370 COBOL for VSE	IBM COBOL for MVS & VM COBOL Visual Set for OS/2 COBOL Set for AIX
			Object - Oriented Extensions (emerging '97 Std)
		Intrinsic Functions (addendum t '85 Std) Language Extensions Support for LE/370 Support for CODE/370	Intrinsic Functions (addendum t '85 Std) Language Extensions Support for LE/370 Support for CODE/370
	COBOL 85 Standard (no intrinsic functions) Structured Programming DBCS	COBOL 85 Standard (no intrinsic functions) Structured Programming DBCS	COBOL 85 Standard (no intrinsic functions) Structured Programming DBCS
COBOL 74 Standard (line mode)	COBOL 74 Compatibility (line mode)	COBOL 74 Compatibility 85 STD FIPS Flagging	COBOL 74 Compatibility 85 STD FIPS Flagging



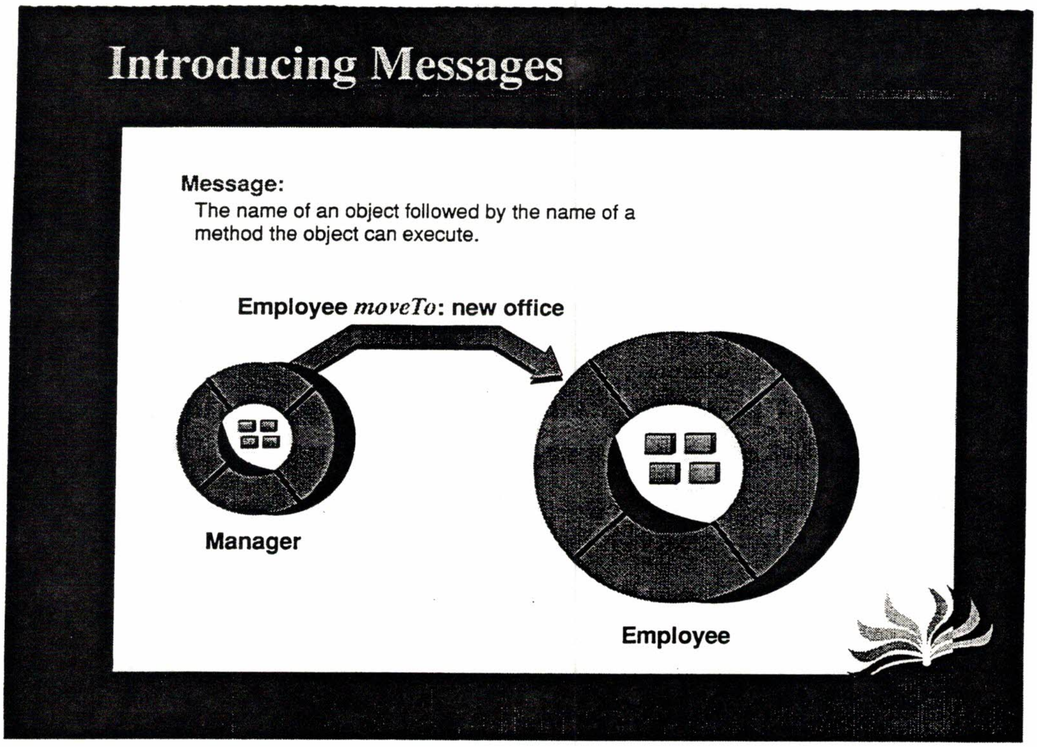
Introducing Objects

Object:

A software "package" that contains a collection of procedures (methods) and data(variables)

Methods

Variables



Introducing Messages

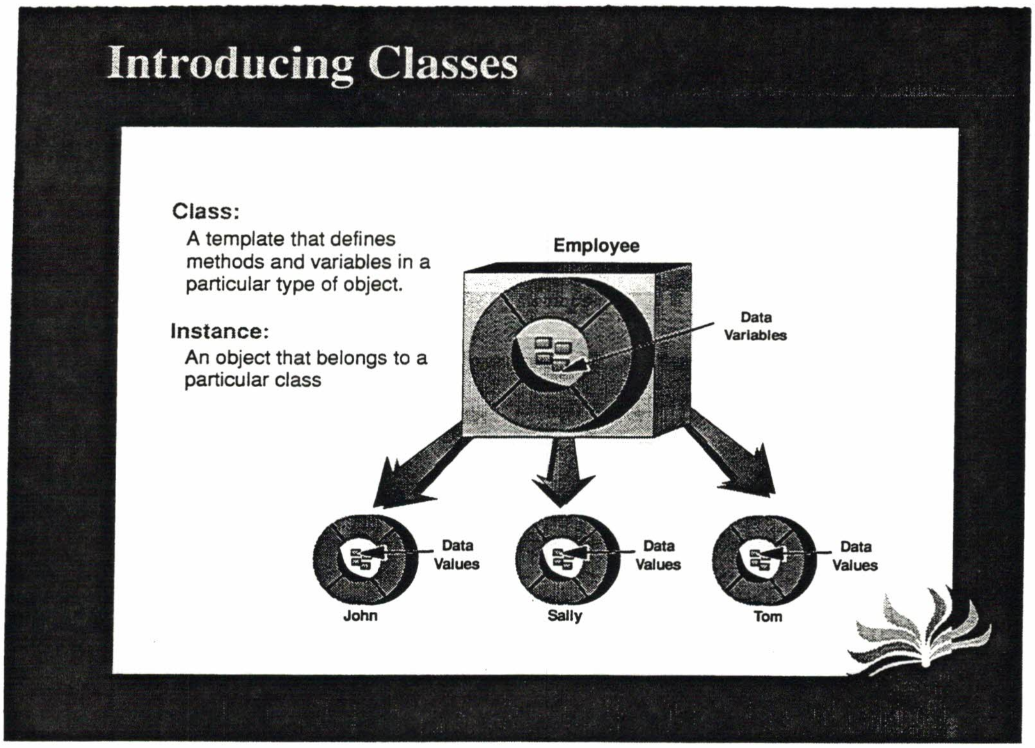
Message:

The name of an object followed by the name of a method the object can execute.

Employee *moveTo*: new office

Manager

Employee

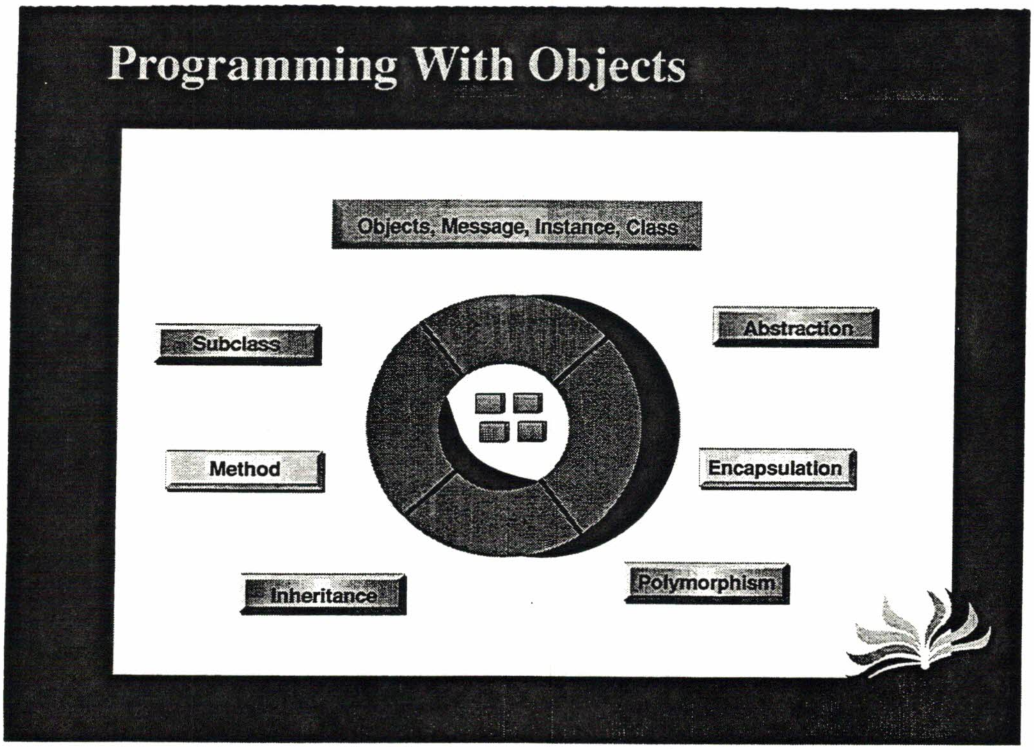


Introducing Classes

Class:
A template that defines methods and variables in a particular type of object.

Instance:
An object that belongs to a particular class

Employee
Data Variables
Data Values
Data Values
Data Values
John
Sally
Tom



Programming With Objects

Objects, Message, Instance, Class

Subclass

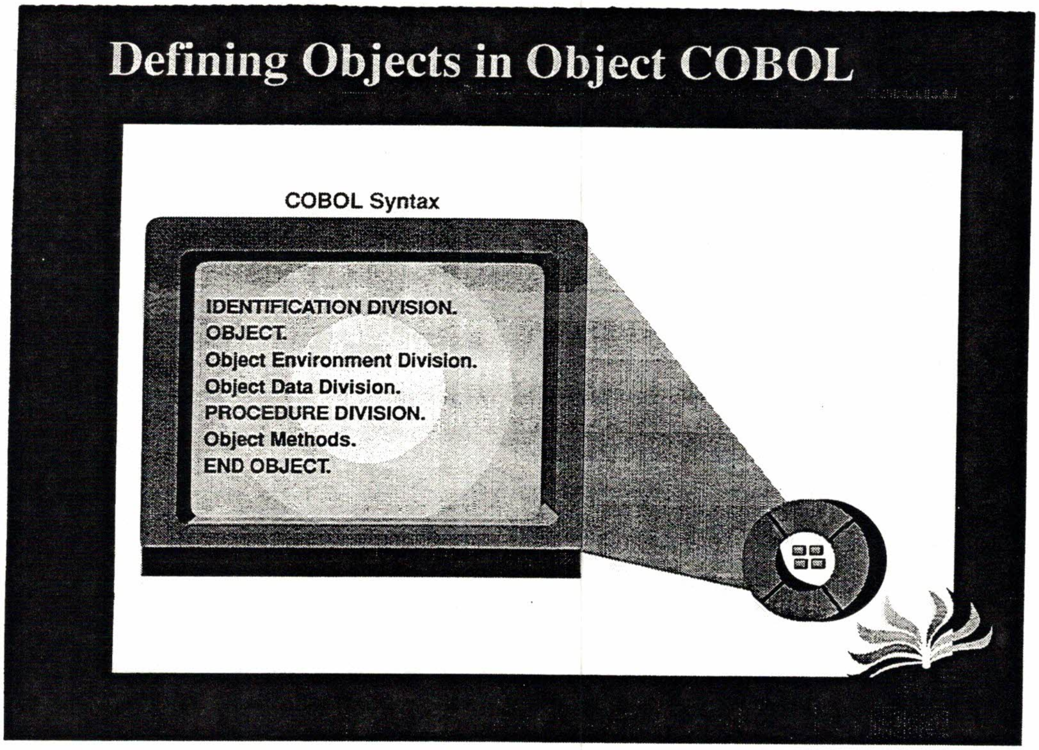
Abstraction

Method

Encapsulation

Inheritance

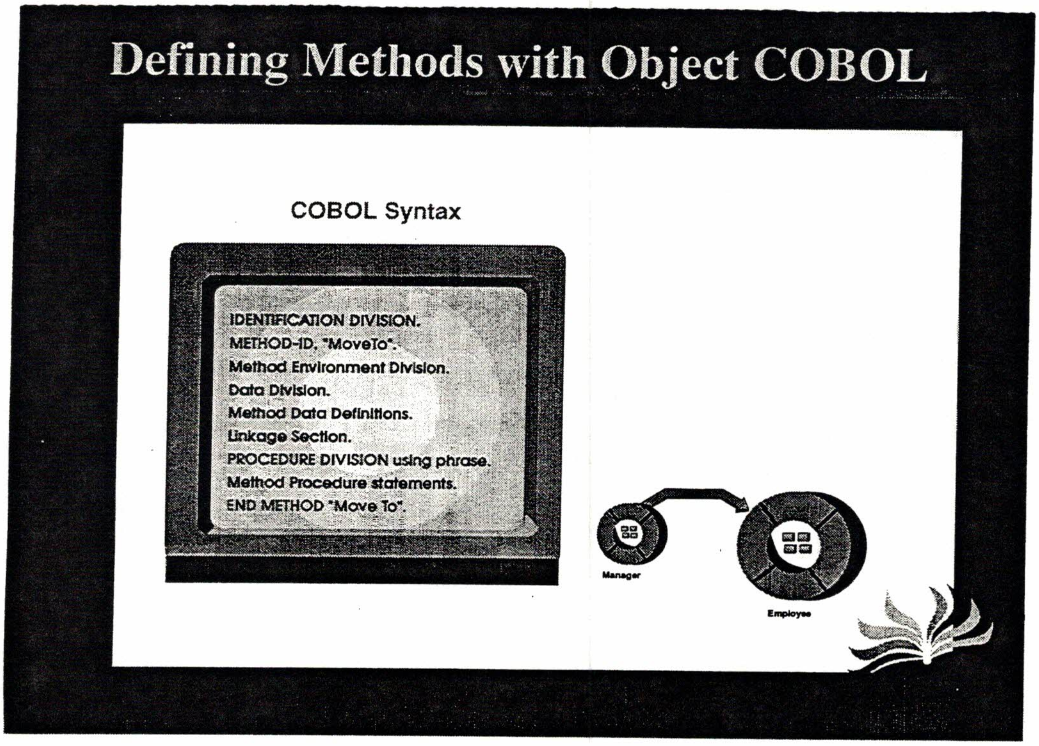
Polymorphism



Defining Objects in Object COBOL

COBOL Syntax

IDENTIFICATION DIVISION.
OBJECT.
Object Environment Division.
Object Data Division.
PROCEDURE DIVISION.
Object Methods.
END OBJECT.

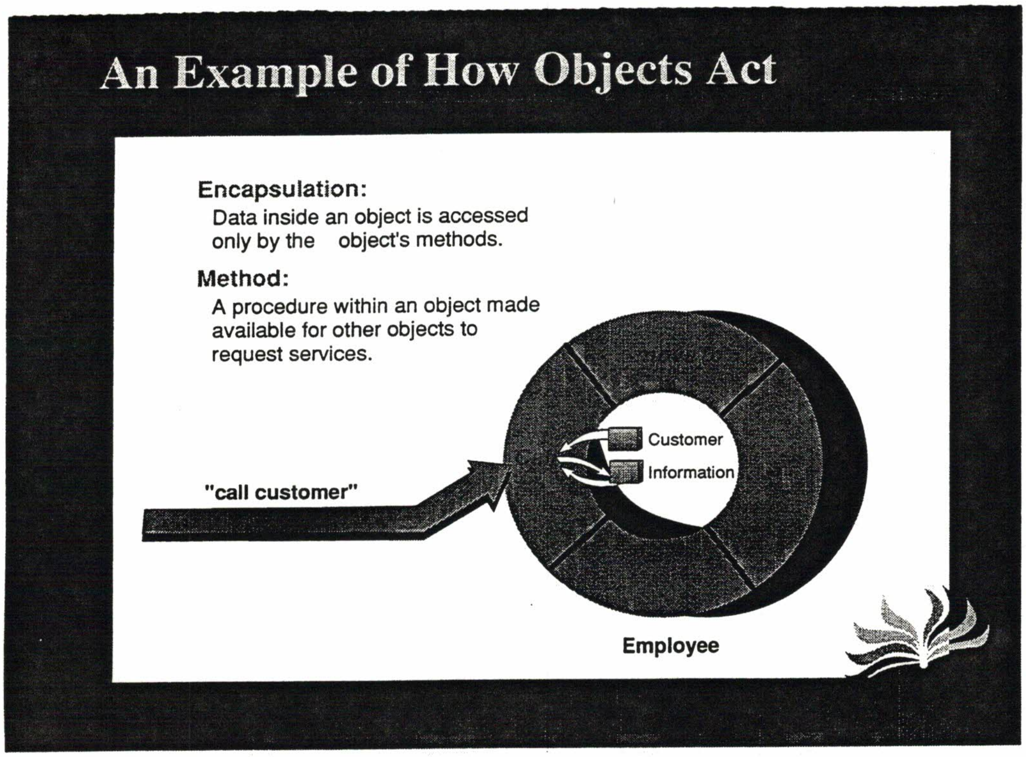


Defining Methods with Object COBOL

COBOL Syntax

IDENTIFICATION DIVISION.
METHOD-ID. "MoveTo".
Method Environment Division.
Data Division.
Method Data Definitions.
Linkage Section.
PROCEDURE DIVISION using phrase.
Method Procedure statements.
END METHOD "Move To".

Manager
Employee

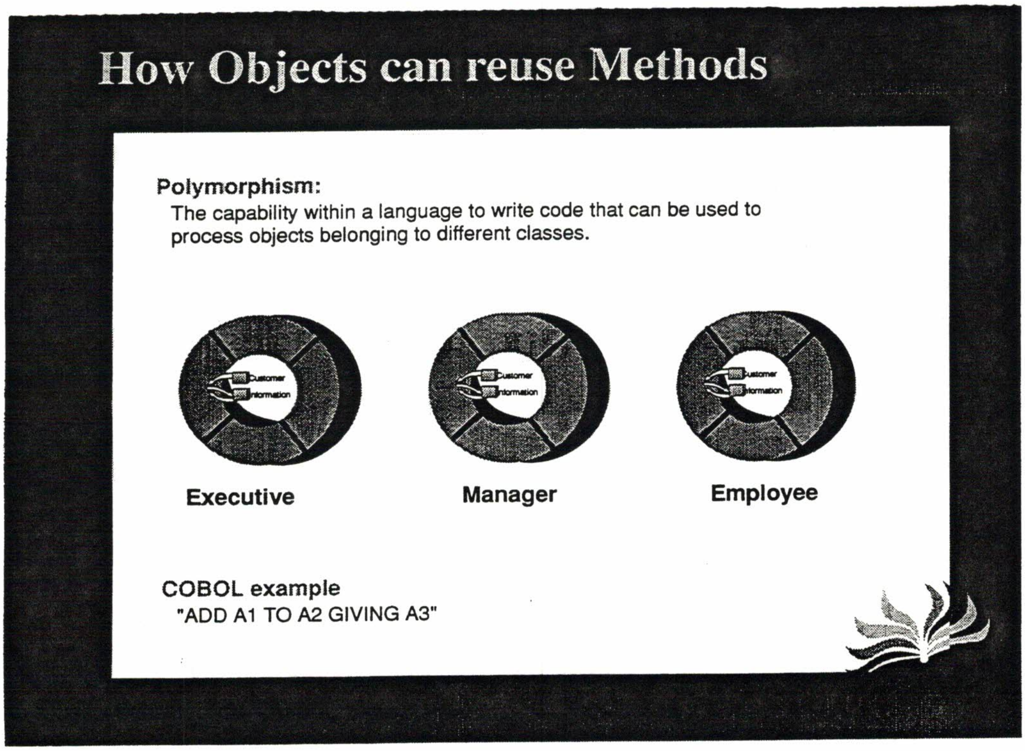


An Example of How Objects Act

Encapsulation:
Data inside an object is accessed only by the object's methods.

Method:
A procedure within an object made available for other objects to request services.

Customer
Information
"call customer"
Employee

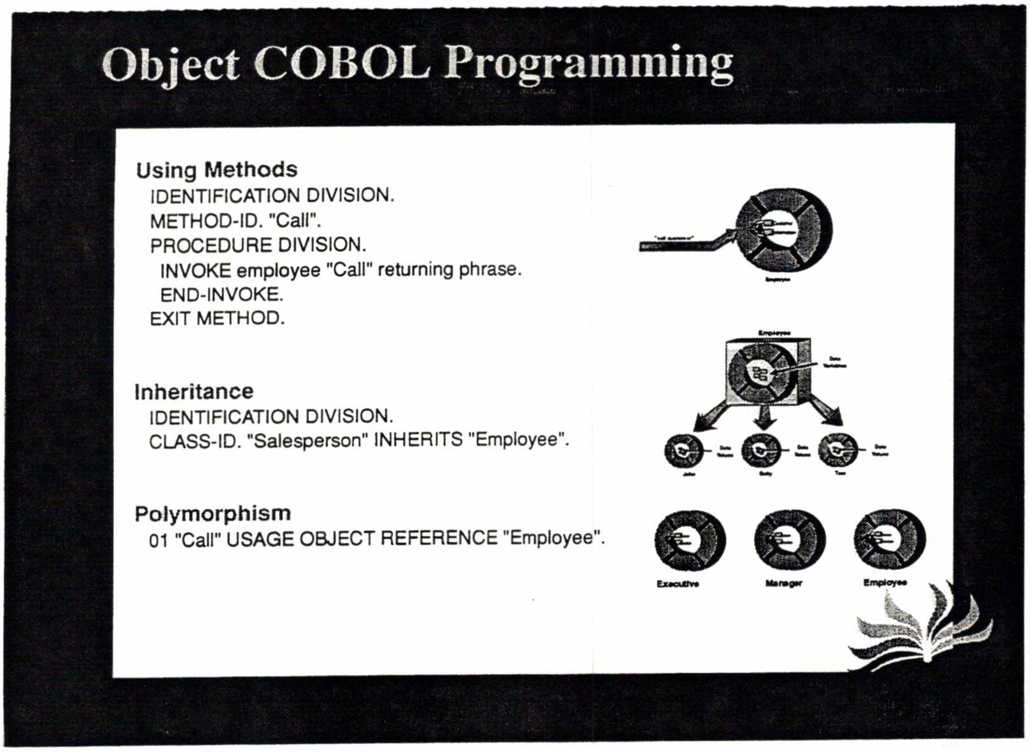


How Objects can reuse Methods

Polymorphism:
The capability within a language to write code that can be used to process objects belonging to different classes.

Customer
Information
Executive
Manager
Employee

COBOL example
"ADD A1 TO A2 GIVING A3"



Object COBOL Programming

Using Methods

```
IDENTIFICATION DIVISION.
METHOD-ID. "Call".
PROCEDURE DIVISION.
 INVOKE employee "Call" returning phrase.
 END-INVOKE.
EXIT METHOD.
```

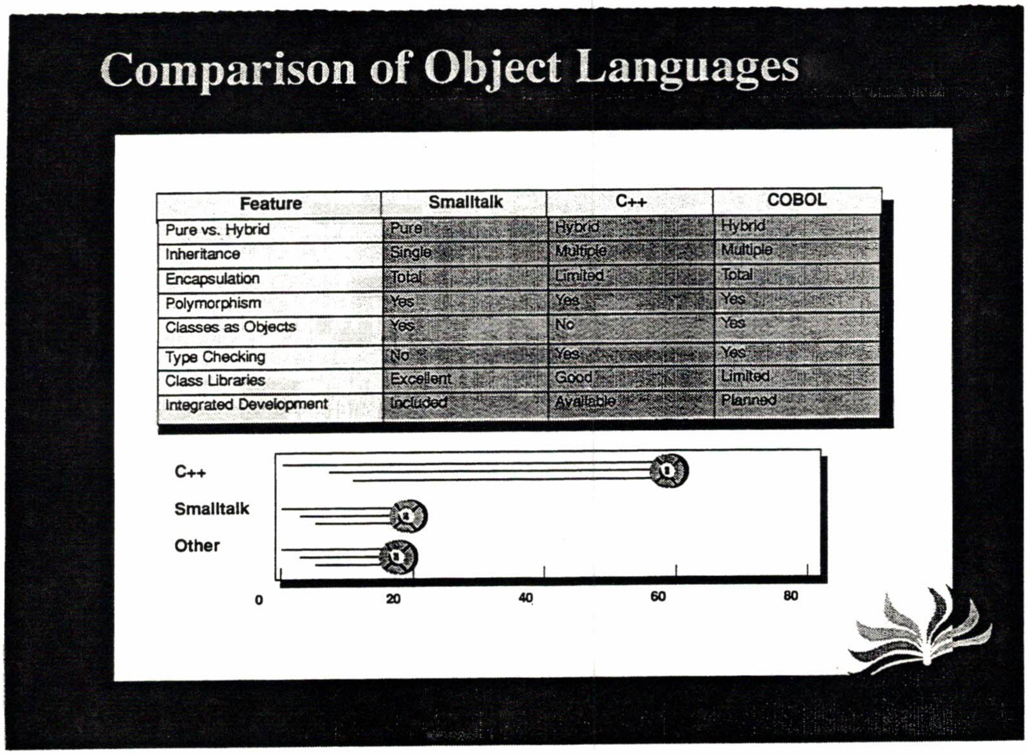
Inheritance

```
IDENTIFICATION DIVISION.
CLASS-ID. "Salesperson" INHERITS "Employee".
```

Polymorphism

```
01 "Call" USAGE OBJECT REFERENCE "Employee".
```

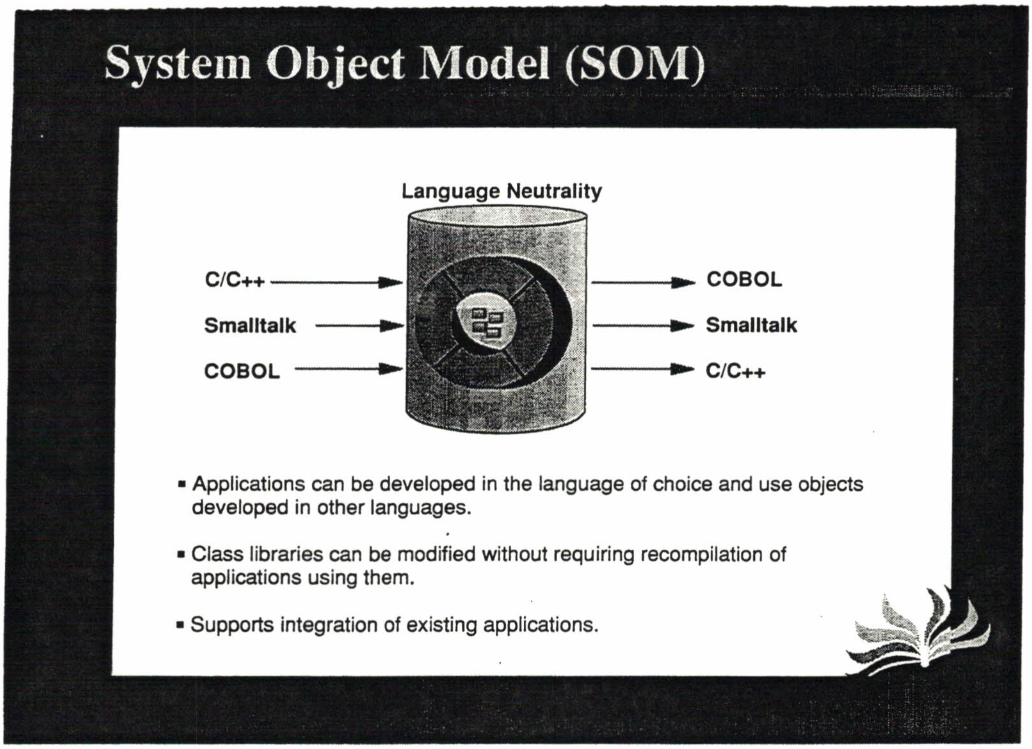
Employee
Executive
Manager
Employee



Comparison of Object Languages

Feature	Smalltalk	C++	COBOL
Pure vs. Hybrid	Pure	Hybrid	Hybrid
Inheritance	Single	Multiple	Multiple
Encapsulation	Total	Limited	Total
Polymorphism	Yes	Yes	Yes
Classes as Objects	Yes	No	Yes
Type Checking	No	Yes	Yes
Class Libraries	Excellent	Good	Limited
Integrated Development	Included	Available	Planned

C++
Smalltalk
Other
0
20
40
60
80



System Object Model (SOM)

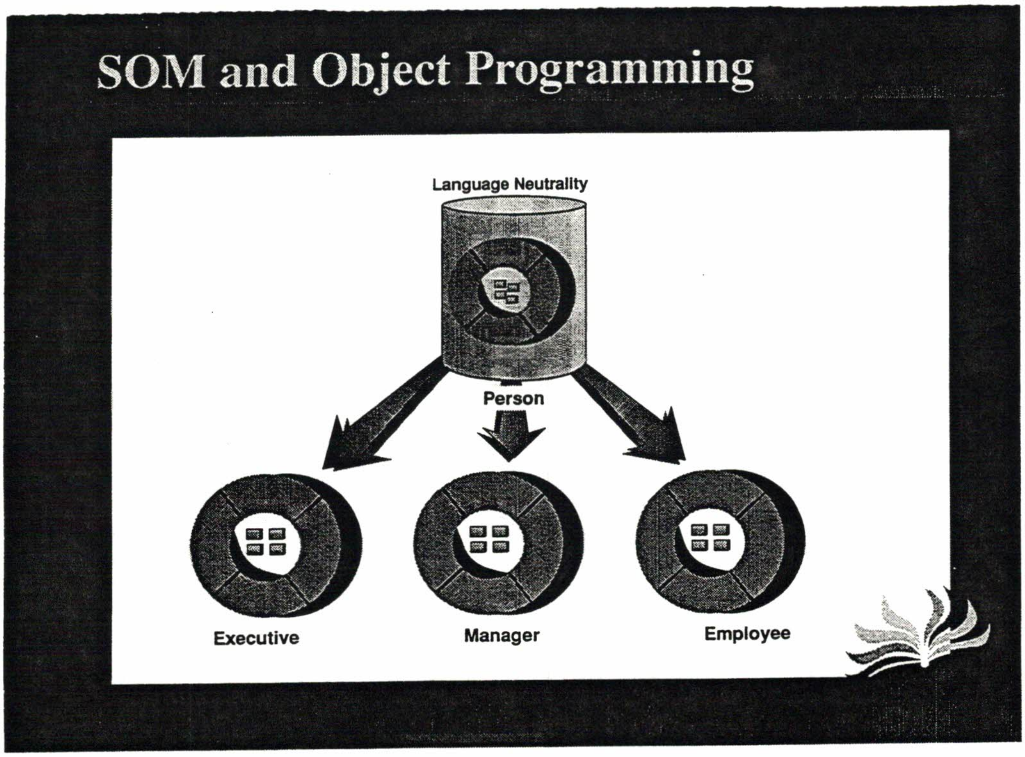
Language Neutrality

C/C++ → COBOL

Smalltalk → Smalltalk

COBOL → C/C++

- Applications can be developed in the language of choice and use objects developed in other languages.
- Class libraries can be modified without requiring recompilation of applications using them.
- Supports integration of existing applications.



SOM and Object Programming

Language Neutrality

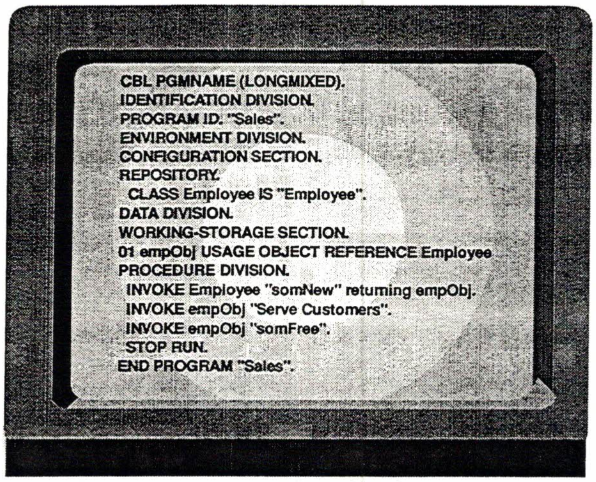
Person

Executive

Manager

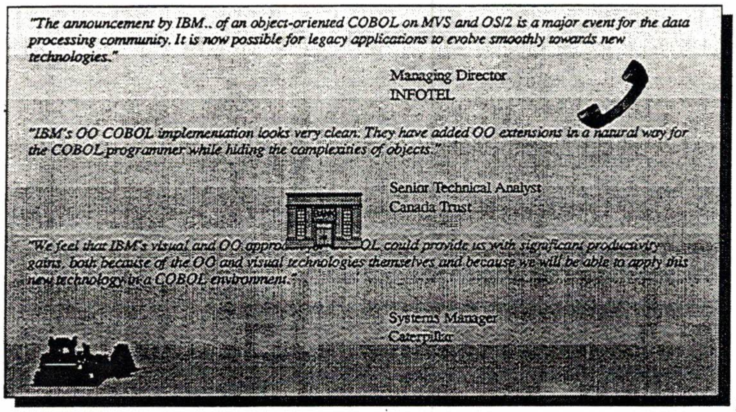
Employee

IBM COBOL with SOM



```
CBL PGMNAME (LONGMIXED).
IDENTIFICATION DIVISION.
PROGRAM ID. "Sales".
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
REPOSITORY.
 CLASS Employee IS "Employee".
DATA DIVISION.
WORKING-STORAGE SECTION.
01 empObj USAGE OBJECT REFERENCE Employee
PROCEDURE DIVISION.
 INVOKE Employee "somNew" returning empObj.
 INVOKE empObj "Serve Customers".
 INVOKE empObj "somFree".
 STOP RUN.
END PROGRAM "Sales".
```

What Customers are Saying



"The announcement by IBM.. of an object-oriented COBOL on MVS and OS/2 is a major event for the data processing community. It is now possible for legacy applications to evolve smoothly towards new technologies."

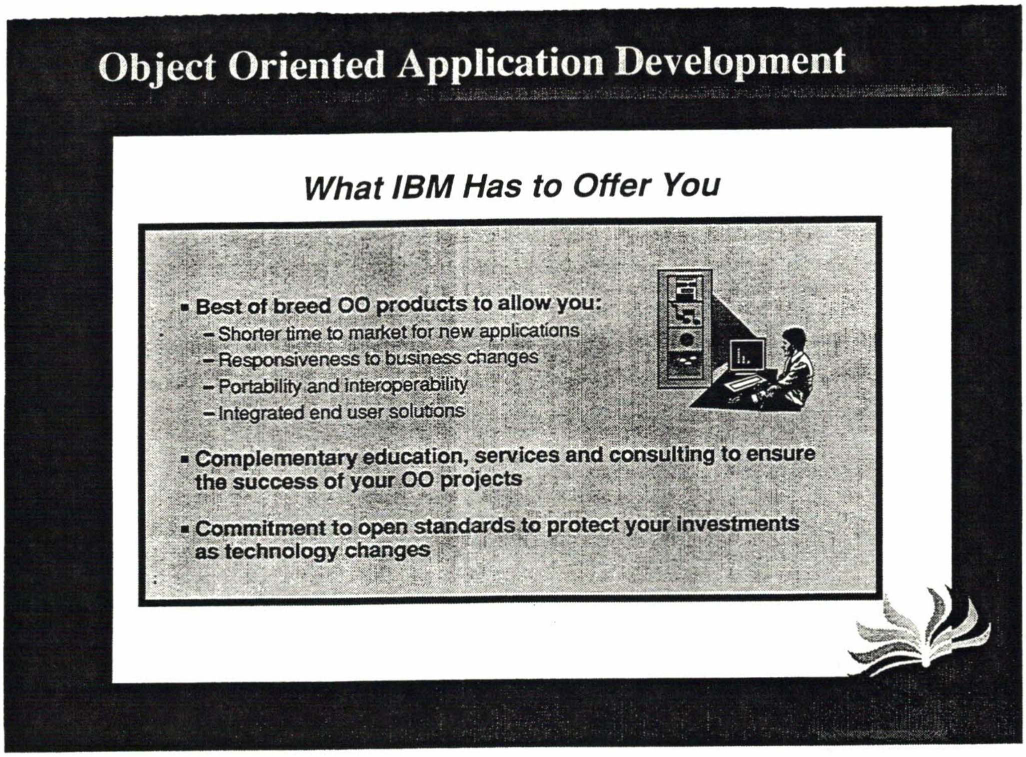
Managing Director
INFOTEL

"IBM's OO COBOL implementation looks very clean. They have added OO extensions in a natural way for the COBOL programmer while hiding the complexities of objects."

Senior Technical Analyst
Canada Trust

"We feel that IBM's visual and OO appro [illegible] OL could provide us with significant productivity gains, both because of the OO and visual technologies themselves and because we will be able to apply this new technology in a COBOL environment."

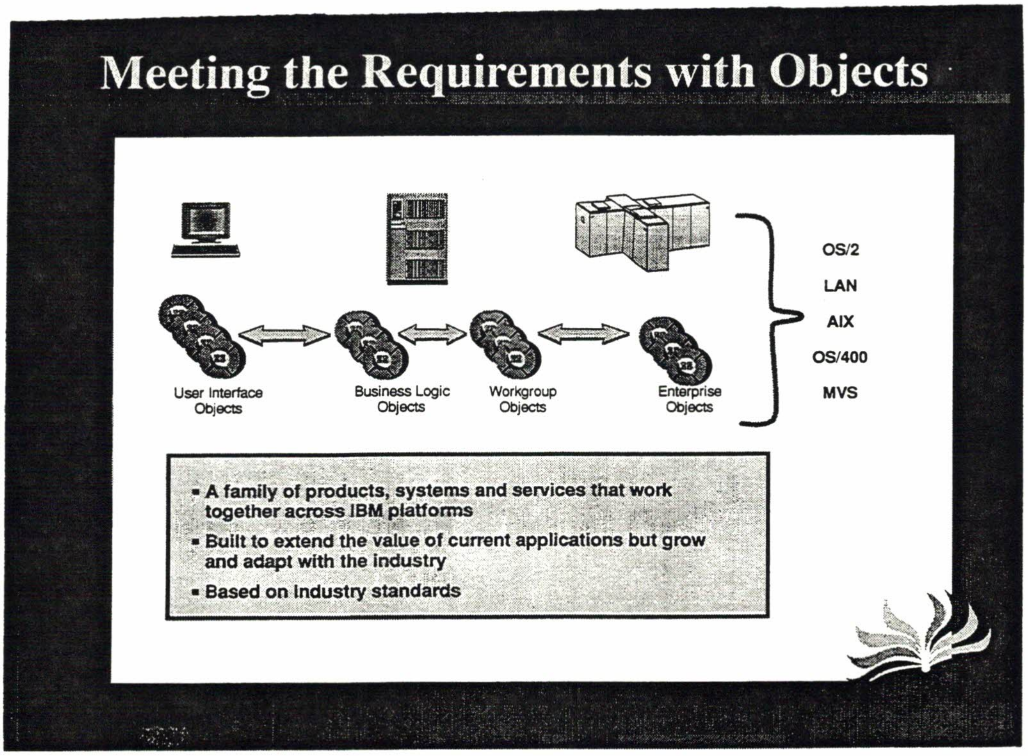
Systems Manager
Caterpillar



Object Oriented Application Development

What IBM Has to Offer You

- **Best of breed OO products to allow you:**
  - Shorter time to market for new applications
  - Responsiveness to business changes
  - Portability and interoperability
  - Integrated end user solutions
- **Complementary education, services and consulting to ensure the success of your OO projects**
- **Commitment to open standards to protect your investments as technology changes**



Meeting the Requirements with Objects

User Interface Objects
Business Logic Objects
Workgroup Objects
Enterprise Objects
OS/2
LAN
AIX
OS/400
MVS

- **A family of products, systems and services that work together across IBM platforms**
- **Built to extend the value of current applications but grow and adapt with the industry**
- **Based on industry standards**

PRO 0511

SOMobjects: Management Utilities for Distributed SOM

Bart Jacob

IBM

Austin, TX

Trademarks The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX IBM OS/2 SOMobjects

The following terms are trademarks of other companies:

Microsoft, Windows	Microsoft Corporation	OMG	Object Management Group

Other trademarks are trademarks of their respective companies.

The term "CORBA" used in this book refers to the Common Object Request Broker Architecture standards promulgated by the ObjectManagement Group, Inc.

ABSTRACT

Most software development organizations are currently moving towards the use of object-oriented technology to speed program development and to reap the benefits associated with code reuse this technology enables. At the same time, networking capabilities and the requirement for client/server applications are growing. The logical result of these two trends is the requirement to provide support for distributed objects. The Distributed SOM framework built on the System Object Model (SOM) technology provides this environment.

However, as in all distributed environments, there are configuration and management issues related to distributing objects across processes and systems. Though the DSOM framework in SOM V2.1 provides some APIs and facilities to assist with these management requirements, there are few utilities that an administrator can use to understand the current state of the environment and to alter this state as required. Additionally, there is a requirement during program development and testing to understand and monitor the state of the DSOM environment.

This paper overviews a series of utilities which were written to fulfill some of the requirements for monitoring and managing the Distributed SOM environment. Most of these utilities were written to documented interfaces already provided by SOM and DSOM. Though many of these utilities may be used as written, they are intended to be used as samples upon which developers may build to provide specific functionality to meet their requirements.

This paper is extracted from the IBM Technical Bulletin (Redbook): SOMobjects: Management Utilities for Distributed SOM, GG24-4357. The technical bulletin includes a diskette with the source code for the various utilities described within.

RELATED PUBLICATIONS

If you are interested in learning more about DSOM you may be interested in the following publications.

- SOMobjects: A Practical Introduction to SOM/DSOM, IBM Order No. - GG24-4357 Object-Oriented Programming Using SOM and DSOM, (Lau) ISBN 0-442-01948-3 Objects for OS/2, (Danforth, Koenen, Tate) ISBN 0-442-01738-3 SOMobjects Toolkit Documentation, (Softcopy with Product) SOMobjects Workgroup Enabler Installation/Configuration Guide for OS/2, AIX and Windows, (Hardcopy with Product)

IBM Publications may be ordered by Phone: 800-879-2755 or by Fax: 800-284-4721

For a list of Object Management Group publications or additional information on OMG, please contact: +1-508-820-4300, fax +1-508-820-4303, or email: info@omg.org

INTRODUCING SOM AND DSOM

The System Object Model (SOM) is a technology for the packaging of software objects. This technology is unique in that it provides:

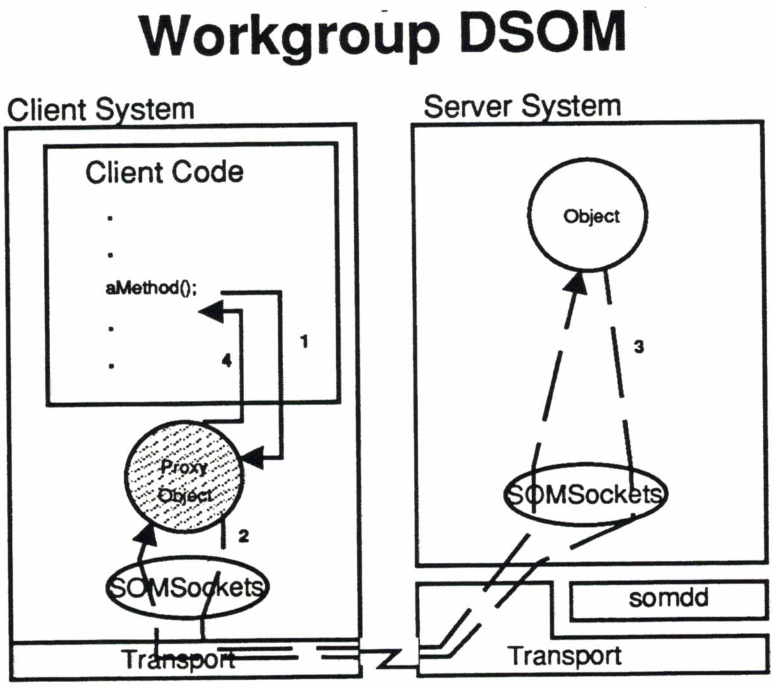
- State-of-the-art object technology
- Programming language independence
- Release-to-release binary compatibility of class libraries

The DSOM framework is a set of SOM classes which provide a CORBA compliant ORB for the distribution of SOM objects. Currently, SOM and DSOM are supported on OS/2, AIX and Microsoft Windows-based systems.

DSOM Implementation Basics

DSOM provides for cross-process access of SOM objects. These processes can reside on the same or different physical systems.

The following figure provides a general view of the DSOM environment.



There are a minimum of three processes involved. The client process includes the application logic that will require access to an object. Methods are invoked on proxy objects which reside in the same process as the client. As a result of invoking a method on a proxy object, the method invocation message is passed to the second process, where the actual object exists. The method is then executed and the results are passed back to the client via the same mechanism.

The third required process is running the DSOM daemon (an executable called 'somdd'). This daemon tracks the existence of server processes and will start new server processes as required. The DSOM daemon is required to be active for the clients and servers to communicate with one another. The DSOM daemon is required on server systems.

DSOM CLIENTS

DSOM clients are programs that will access and perform operations on objects that exist in other processes.

As in any distributed object environment, the following issues must be addressed:

- How does one locate a server process that can support objects of a specific class?
- How does one create an appropriate server process if one does not currently exist?
- How does one create an object in a server process?
- How does one reference programming objects that exist in a different process (and possibly a different system)?
- How does one destroy objects, references to objects and server processes?

There are different aspects as to how the above questions might be answered. On one hand, there must be standard programming interfaces that a client can use to accomplish these tasks. On the other hand, there must also be a process of configuring the DSOM environment so as to hide the actual location of objects and object servers from the programmer.

The primary components of DSOM (from a client perspective) that provide the above capabilities are:

DSOM OBJECT MANAGER

The DSOM Object Manager is an object that will exist in the client process and provide a programming interface (through exposed methods) for the client to locate and gain access to servers and objects. The DSOM Object Manager is an instance of the class SOMDObjectMgr.

PROXY OBJECTS

Proxy objects will be created in the client process to represent the actual objects that will be located in servers. Clients may invoke methods upon the proxy objects as if they were the target object. The implementation of the proxy object will handle the marshalling of the method invocation message and passing it to the server to be executed.

IMPLEMENTATION REPOSITORY

One can think of the Implementation Repository as a set of directory services that provide information regarding possible DSOM server processes. This repository can be populated at system/network configuration time and therefore the programmer does not need to know on exactly what system or in which process a distributed object will exist.

INTERFACE REPOSITORY

The Interface Repository contains information about the object classes and their interfaces. This information must be accessed by the DSOM run-time components in a client process to properly build object proxies and to marshal the method calls for remote objects.

CLIENT SUMMARY

In general, the programmer need only interface with the DSOM Object Manager. The implementation of this object utilizes the Implementation and Interface Repositories and will return proxy objects as required.

The DSOM administrator must will be responsible for populating the Implementation and Interface Repositories with the appropriate information.

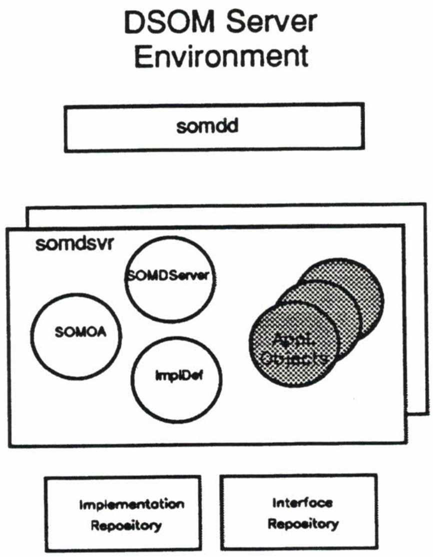
DSOM SERVERS

Systems that will actually contain the object implementations of distributed objects in a DSOM environment are called DSOM servers. DSOM servers will contain at least two different types of processes related to DSOM. First, there will be a single process called the DSOM daemon (an executable called 'somdd'). Second, there will be one or more server processes that will actually contain the objects to be accessed from a DSOM client.

On any system that will take on the role of a DSOM server, a special program called 'somdd' must be running. The program, referred to as the DSOM daemon, tracks the existence of server processes on the system where it exists. In addition, the DSOM daemon will create a new process if required to fulfill a request from a client.

SERVER PROCESSES

In DSOM, the process that manages a target object is called the object's server. Servers are implemented as programs that use SOM classes. The DSOM server environment is depicted in the following figure:



A server can be thought of as consisting of three parts:

- A main program that provides an address space for the objects it manages and one or more threads that can execute method calls. The default server, 'somdsvr', can be replaced by a user written program.

- A set of DSOM framework objects. The primary object being derived from the SOMDServer class which provides methods used to manage objects within the server process. In addition, other objects such as the SOM Object Adapter (SOMOA) object and the Implementation Definition (ImplementationDef) object also exist in the server process. They are designed to work with the SOMDServer object to provide the function required of a DSOM server.

- One or more class libraries which provide the object implementations. These libraries are usually constructed as dynamic link libraries (DLLs) so they can be loaded and linked by a server program dynamically.

The server processes must also have access to the Implementation and Interface Repositories.

MANAGING DSOM RESOURCES

As in most distributed environments, there is a requirement to manage DSOM server systems. This includes server configuration, monitoring, activation, deactivation and so on, preferably from a central location.

SOMobjects Version 2.1 provides a new class to assist with these tasks and a utility to take advantage of the interfaces provided by that class.

The *SOMDServerMgr* class provides methods to:

- Enable a server process so that it can be started when required. Initially, all server processes are enabled by default.
- Disable a server process from starting. Once, a server process is disabled, it cannot be restarted until it is explicitly enabled again.
- Query the status of a server process. The status will indicate whether the server is currently running, is in a disabled state or if the server is not found.
- Stop a server process
- Start a server process

A command line utility called 'dsom' is provided that utilizes this class and provides an administrator with the capability to manage the DSOM servers within the network.

ADDITIONAL MANAGEMENT UTILITIES

We took on a project to develop a set of utilities to demonstrate programming techniques for the development of additional DSOM management utilities. This section includes a short overview of the utilities we developed and have made available with our technical bulletin.

SOMDD PING UTILITY

A ping utility ('somdping') that will determine if a DSOM server system is ready to accept requests (that is, 'somdd' is active and ready).

This utility can be used stand-alone, but is also integrated with some of the other facilities described below.

'somdping' is a simple program that runs from a standard command prompt. As its primary parameter, it takes the network name of a system and determines whether 'somdd' is active (or contactable) on that system.

It returns a value of 0 if 'somdd' responds correctly and a value of -1 otherwise. Therefore, this command can be used in command files or shell scripts as one step in a more complex procedure.

ITSCOBJECTMGR CLASS

A subclass of *SOMDObjectMgr*. In the current DSOM implementation, a client program attempting to access a server must wait for SOMDTIMEOUT seconds before being notified that the desired server is not available. The *ITSCObjectMgr* class provides new methods such as `FindActiveServerBy....()`, which utilize the class underlying 'somdping', to determine immediately whether a server system's 'somdd' is in a ready state. We have provided a text-based client which utilizes this class and demonstrates how it may be used within client programs you may develop.

The *ITSCObjectMgr* is derived from the *SOMDObjectMgr* and therefore provides all of the functionality provided by its parent class. We have not overridden any of the methods of *SOMDObjectMgr* except for `somInit()`.

We have introduced the following methods:

- `somdFindAnyActiveServerByClass()`
- `somdFindActiveServersByClass()`
- `isServerActive()`
- `somdFindAllServersOnHost()`
- `somdFindAllClassesOnHost()`
- `somdFindAllHostsByClass()`

The first two methods, utilize the *PingSdd* class (which underlies somdping), to locate DSOM servers on whose systems '`somdd`' is currently active. A server proxy that is returned by these methods, does not mean the server process itself is necessarily currently active, but only that the system on which it will exist has a currently active DSOM daemon.

The third method, `isServerActive()`, tests whether the system for which a particular proxy already exists, has an active DSOM daemon. Therefore, one could use one of the original *SOMDObjectMgr* methods to obtain a proxy to a server object and then use **isServerActive()** to check on the status of the server system before attempting to invoke a method on that server.

The last three methods, provide simplified ways of querying the Implementation Repository to determine what server implementations or classes are supported by a specific host and what hosts are registered with the Implementation Repository as supporting specific classes.

ITSCSOMDSERVER CLASS

A subclass of *SOMDServer* that provides additional methods to trace the activities in a server and to check the status of the objects in the server. This class is utilized by some of the other management programs we are providing.

We have provided a text-based driver program to test and demonstrate the functionality of this class.

ITSCSOMDServer is derived from *SOMDServer* and introduces the following methods:

```
GetAllRegisterClasses()
GetAllUserClasses()
GetAllAffinityGroup()
GetClassStatus()
GetClassTree()
GetClassAncestors()
```

```
GetInstancesForClass()
GetRefData()
verifyList()
```

ITSCSVR PROGRAM

A replacement for the default server program, 'somdsvr' provides a GUI interface to an active server. This utility also provides dynamic tracing of the methods invoked on objects in the server and displays a wealth of information about the active classes and objects in the server. This program takes advantage of the ITSCSOMDServer class.

The program provides a GUI through which one can:

- View the server implementation definition The server implementation definition found in the Implementation Repository is displayed by the '`itscsvr`' user interface. This provides immediate information as to what server implementation is running in the current process.

- Query information regarding loaded classes This facility shows what class objects are currently loaded within the server process. One can also view instances of these classes with information about which client created them.

This facility is only available if the server class object is an instance of *ITSCSOMDServer* or an instance of a class derived from *ITSCSOMDServer*.

- Dynamically start and stop tracing
A dynamic trace facility is provided to allow the user to view activity within the server or to display the output of `somPrintf()` statements within classes running in the server.

The **Trace** notebook page within the main window displays all output generated by `somPrintf()` function calls in the server process. In addition, this also includes SOM trace output controlled by the global variables:

```
 - SOMD_DebugFlag
 - SOM_TraceLevel
- SOM_WarnLevel
- SOM_AssertLevel
```

These global variables can by dynamically changed through the **Trace** option on the menu bar.

In addition, the *ITSCSOMDServer* class provides another trace option (`SOMD_DispatchLevel`) that can also be controlled through the '`itscsvr`'.

If using an instance of *ITSCSOMDServer*, a secondary window is available to view the loaded classes in the server for obtaining information regarding these classes.

SVRMGR PROGRAM

This program provides a GUI-based utility for monitoring and changing the status of DSOM servers in the network. They utilize the *SOMDServerMgr*, the *ITSCSOMDServer* class, as well as the class underlying the 'somdping' utility.

These programs allow an administrator to start, stop, enable, disable and query DSOM servers within a network.

The application consists primarily of a main container window. Within this window, icons represent the various DSOM servers defined in the Implementation Repository. The '`svrmgr`' allows an administrator to perform the following functions from a central workstation:

- Add a server definition to the Implementation Repository
- Update the server definition for an existing server
- Delete the server definition for an existing server
- Explicitly start a remote server process
- Stop a remote server process
- Query the status of a server process
- Enable the server so it can be started when required
- Disable the server process from starting
- Display the current object references existing in a server

ITSCIR PROGRAM

'`itscir`' was developed to provide a graphical view of the Interface Repository. It provides a hierarchical view of the class relationships within the Interface Repository and allows the Interface Repository data on an individual class level to be easily referenced.

The '`itscir`' utility provides a graphical user interface with a single main window. The main window is partitioned three ways.

The first partition lists all the classes defined in the Interface Repository and provides visual information regarding their hierarchy. The second partition displays the methods and attributes supported by a selected class. The third partition will show either the '.idl' file or the the '.c' or '.cpp' implementation for a selected class. The source and '.idl' files must be able to be found by searching an appropriate path.

MISCELLANEOUS OS/2 COMMAND FILES

Several sample OS/2 command files were developed that assist with environment checking, and starting and stopping the DSOM daemon and servers. These command files are meant as simple examples of what can be done. Similar shell scripts could easily be written for an AIX environment.

SUMMARY

This paper has provided a very high-level view of the DSOM environment and has touched on some of the considerations for managing this environment. We could not go into specific details of DSOM or its management facilities in such a short paper.

The DSOM framework does provide several facilities for assisting with management and administration. However, it also provides a rich set of interfaces through supplied classes that may be used directly or subclassed to build powerful management applications. A set of sample applications demonstrating these capabilities is available through the document on which this paper was based and which is mentioned in the abstract.

this page intentionally left blank

SYS 0505

Understanding OS/2's Multiprotocol Interoperability

Bob Panza

The Systems Expert, Inc.
Woodbury, Conn.

SYNOPSIS

Multiprotocol options, IBM-Novell and Novell-IBM interoperability, and numerous configurations will be reviewed in detail for the latest version of the software (NetWare Client, v2.1 and OS/2, v2.11 at the time of this writing). Practical application of interoperability will be discussed. This session is part of a series on the NetWare Client for OS/2 which includes #086, #093, and #101.

AGENDA

1. Multiprotocol overview
2. An indepth definition of interoperability
3. IBM on Novell's ODI: ODINSUP diagram and details ofthe configuration files, including NET.CFG,*config.sys*, and *protocol.ini*
4. Novell on IBM: LANSUP diagram and details of theconfiguration files, i.e., NET.CFG, *config.sys*, and*protocol.ini*
5. Novell on IBM (NDIS): ODI2NDI with details of theconfiguration files, e.g., NET.CFG, *config.sys*, and*protocol.ini*
6. IBM NTS/2 LAPS tuning overview
7. Dualing NetBIOS
8. Multiple protocol implementation
9. Review and explanation of NetWare Client architecturediagrams in OS2DOC-1.

UNDERSTANDING OS/2'S MULTIPROTOCOL INTEROPERABILITY

A network board may be shared with IBM Extended Services, LAN Server, and LAN Requester. We will focus our interoperability illustrations on Extended Services (now Communications Manager/2) for the sake of "simplicity". Nevertheless, interoperability provisions for Extended Services also form the basis required for joining the NetWare Client with IBM LAN Services.

NOVELL INTEROP

Two NetWare Client for OS/2 drivers facilitate the implementation: *odinsup.sys*and LANSUP.SYS. *odinsup.sys*allows Extended Services and LAN Services to use Novell's ODI network driver. In that case, the board is controlled and configured by *odi.lansup.sys* does the reverse. It enables the NetWare Client to use Extended Services (which retain control of the network board). IBM offers another approach, via Network Transport Services/2 (NTS/2), to give Novell some "coopetition".

IBM communications software and NetWare Client software must be installed on the workstation in order to have all of the pieces in place prior to operation over the network. The NetWare Client is installed and configured via the INSTALL program that comes on NetWare OS/2 Client, v2.01 floppies. Refer to "NetWare Made Easy" in the February 1994 issue of OS/2 magazine for a cook book approach to installation. The IBM software may utilize a LAN Adapter and Protocol Support (LAPS) program for installation and configuration of the appropriate communication files and interfaces. LAPS is part of the IBM NTS/2 package residing in the \IBMCOM tree. NTS/2 is analogous to the IBM LAN Support Program, which provides low layer communication interfaces for workstations running DOS.

Software installation works easiest if the software to control the adapter is installed last. In an ODINSUP configuration, IBM software should be installed before the NetWare Client. For implementation of LANSUP or

ODI2NDI, the NetWare Client software should be installed prior to the IBM software.

Implementation of *odinsup.sys*impacts the *protocol.ini* and *config.sys* as well as the *net.cfg*file. The objective in configuring for ODINSUP is to remove the Extended Edition interface at the Data Link MAC sublayer and to slide an ODI interface in its place.

Within the *protocol.ini* file resident in \IBMCOM, all NDIS MAC driver lines need to be remarked out or deleted. For example, the IBM Token-Ring adapter NDIS MAC driver line is BINDINGS IBMTOK-NIF [- underscore]. The line may be found in the *protocol.ini* file within the LANDD-nif section (and NETBEUI-nif section if LAN Services is configured [- underscore]). The new bindings line will be BINDINGS TOKEN, assuming that we want to bind NDIS to the *odi token.sys*driver of the NetWare Client. This new line must be entered into each section of *protocol.ini* where the NDIS MAC driver was found.

A second IBM Token-Ring board could be bound via BINDINGS TOKEN2. The same logic applies to Ethernet adapters, e.g., BINDINGS NE2000 replaces the NDIS MAC driver lines throughout the *protocol.ini* file for the Novell NE2000 board. At the end of the *protocol.ini* file, an empty header must be added for each ODI driver. The header would be [TOKEN], for TOKEN.SYS, preceded and followed by blank lines.

Next, the *config.sys* file must be correspondingly changed to disable the NDIS MAC driver. Using Token-Ring as an example again, the IBMTOK.OS2 driver line, DEVICE C:\IBMCOM\MACS\ IBMTOK.OS2, must be remarked or deleted. *odinsup.sys*will need to be entered into the NetWare Requester section of the *config.sys* file following *token.sys*.

*Route.sys*will most likely be required to communicate across bridged Token-Rings. Note that *route.sys*is not necessary if the workstation is on a routed ring since the router will place the source routing information into the workstation's packets when necessary.

Our *net.cfg*changes enable all frame types for Token-Ring or Ethernet as shown below.

```
LINK DRIVER TOKEN
FRAME TOKEN-RING
FRAME TOKEN-RING_SNAP [- before SNAP is an underscore]

LINK DRIVER NE2000
FRAME ETHERNET_802.3
FRAME ETHERNET_802.2
FRAME ETHERNET_II
FRAME ETHERNET_SNAP
```

The first frames listed in each section above are used by the workstation when generating GET NEAREST SERVER broadcasts and must correspond to the frame type generally used by default for SPX/IPX traffic throughout the enterprise. Otherwise, the GET NEAREST SERVER will not attract the appropriate GIVE NEAREST SERVER responses from NetWare file servers.

Since ODI retains control of the adapter, the other *net.cfg*parms are particularly important under the ODINSUP configuration. Our parm tuning exercise will now pay off big dividends as we maximize performance and functionality. Page 3 Lastly, ODINSUP must be bound to the ODI adapter driver to complete the working *net.cfg*-configuration:

```
PROTOCOL ODINSUP
BIND TOKEN
```

or

```
PROTOCOL ODINSUP
BIND NE2000
```

CHANGES FOR ODINSUP CONFIGURATION

* * * PROTOCOL.INI * * *

```
[LANDD.nif]
    DRIVERNAME = LANDD$
;   BINDINGS  IBMTOK-nif [-  underscore]
    BINDINGS  TOKEN
    LINKS  41
    USERS  4MAX-SAPS  5 [-  underscore]
    NETADDRESS  "T400010000BAB

[IBMTOK-nif]
    DRIVERNAME = IBMTOK$

[TOKEN]
```

CAPTION - The IBMTOK-nif bindings line is remarked via ";" in order to deactivate the IBM Token-Ring MAC layer driver. The TOKEN bindings line below it replaces IBMTOK-nif in the stack to facilitate ODI downward to the adapter. The locally administered address (LAA) in NETADDRESS requires a corresponding NODE ADDRESS 400010000BAB entry in the LINK DRIVER section of NET.CFG. If the adapter's "burned-in" address was used rather than an LAA, then the NODE ADDRESS entry in the *net.cfg*would not be required.

* * * CONFIG.SYS * * *

PATH LINES

```
LIBPATH.;C:\OS2\DLL;C:\MUGLIB\DLL;C:\OS2\MDOS;C:\CMLIB\DLL;
C:\;C:\OS2\APPS\DLL;C:\IBMCOM\DLL;C:\NETWARE;

SET PATHC:\OS2;C:\MUGLIB;C:\OS2\SYSTEM;C:\OS2\MDOS\WINOS2;
C:\CMLIB;C:\CMLIB\APPN;C:\OS2\INSTALL;C:\;C:\OS2\MDOS;
C:\OS2\APPS;C:\RUMBAOS2;C:\NETWARE;L:\OS2;P:\OS2;

SET DPATHC:\OS2;C:\MUGLIB\DLL;C:\CMLIB;C:\CMLIB\APPN;
C:\OS2\SYSTEM;C:\OS2\MDOS\WINOS2;C:\OS2\INSTALL;C:\;
C:\OS2\BITMAP;C:\OS2\MDOS;C:\OS2\APPS;C:\IBMCOM;C:\NETWARE;
L:\OS2;
```

CAPTION - IBM Extended Services adds MUBLIB, CMLIB, and IBMCOM directories, and their respective DLL and APPN subdirectories to the *config.sys* path lines as shown above.

IBM COMMUNICATIONS LINES

```
DEVICEC:\IBMCOM\LANMSGDD.OS2 /I:C:\IBMCOM
DEVICEC:\IBMCOM\PROTMAN.OS2 /I:C:\IBMCOM
DEVICEC:\IBMCOM\PROTOCOL\LANDD.OS2
DEVICEC:\IBMCOM\PROTOCOL\LANDLLDD.OS2


DEVICEC:\CMLIB\ACSLDLAN.SYS
RUNC:\OS2\EPW.EXE
RUNC:\IBMCOM\PROTOCOL\LANDLL.EXE
RUNC:\IBMCOM\PROTOCOL\NETBIND.EXE
RUNC:\IBMCOM\LANMSGEX.EXE
```

```
rem c:\ibmcom\macs\ibmtok.os2
DEVICEC:\CMLIB\APPN\CMKFMDE.SYS
```

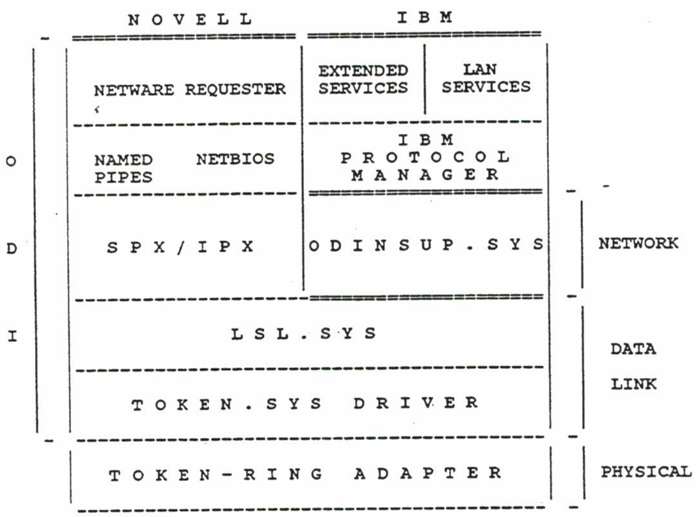
CAPTION - The IBMTOK.OS2 line is remarked out to disable the IBM MAC layer driver. The other IBMCOM and CMLIB entries remain active. They precede the NetWare Requester section of the *config.sys* unless LAN Services is implemented.

NETWARE REQUESTER SECTION

```
SET NWLANGUAGEENGLISH
DEVICEC:\NETWARE\LSL.SYS
RUNC:\NETWARE\DDAEMON.EXE
DEVICEC:\NETWARE\TOKEN.SYS
devicec:\netware\odinsup.sys
DEVICEC:\NETWARE\ROUTE.SYS
DEVICEC:\NETWARE\IPX.SYS
DEVICEC:\NETWARE\SPX.SYS
```

CAPTION - *odinsup.sys*resides in the *config.sys* following the ODI adapter driver, which is *token.sys*in this instance. IBM Protocol Manager precedes the NetWare Requester lines so it will be in place prior the *odinsup.sys*load.

ODINSUP DIAGRAM



CAPTION - For IBM Token-Ring, the Novell ODI stack at the left side of the diagram appears as it would if there was not any IBM software present. IBM Extended Services and LAN Services interface with and utilize the lower layer ODI drivers via the specialized *odinsup.sys*driver. A rough comparison with the lower three layers of the OSI model serves as a guide to this innovative architecture.

SAILING WITH IBM

The Novell driver for interfacing with IBM Extended Services NDIS drivers is *lansup.sys*. It replaces TOKE-NEE.SYS and its successor, CMGRLAN.SYS, from the early days of OS/2 when Extended Edition, v1.X was the IBM OS/2 mainstay. *lansup.sys* was named after LANSUP.COM, the Novell ODI driver that sits above the IEEE 802.2 interface of IBM LAN Support Program on DOS workstations. Likewise, the NetWare OS/2 Client with *lansup.sys* in place runs on top of the IBM Extended Services IEEE 802.2 interface. *lansup.sys* replaces any specific Novell hardware driver, such as *token.sys*or NE2000.SYS. With *lansup.sys*, the full range of frame types do not need to be defined unless the NetWare Requester requires them. Extended Services rather than NetWare will control and therefore configure the adapter. For SPX/IPX traffic, *net.cfg*LINK DRIVER examples applicable to Token-Ring and Ethernet are as follows.

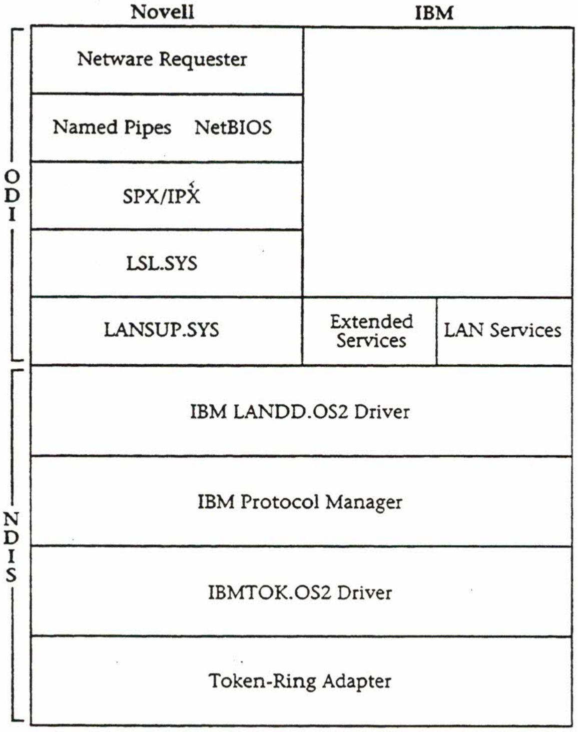
```
LINK DRIVER LANSUP
    FRAME TOKEN-RING
LINK DRIVER LANSUP
    FRAME ETHERNET-802.2 [-  underscore in ETHERNET-]
    FRAME ETHERNET-SNAP  [-  underscore in ETHERNET-]
```

Again, the first FRAME line will be utilized by the GET NEAREST SERVER broadcast. Notice that *lansup.sys* does not support the ETHERNET_802.3 (RAW) frame type. That means trouble for SPX/IPX traffic on enterprise Ethernet. The workaround is to initialize both ETHERNET-802.2 and ETHERNET-802.3 frames on NetWare servers as required. The NetWare servers will then route SPX/IPX traffic between the two logical networks formed by the two different frame types. Every frame type needs its own unique network (NET) number on each side of the router. *net.cfg*parms may be tuned accordingly per NetWare requirements. However, since ODI does not have control of the adapter, the *net.cfg*configuration must not exceed the resources specified within the Extended Services' *protocol.ini* file. The orchestration between *net.cfg*and *protocol.ini* parms is important. For example, the link buffers for IBM Token-Ring 16/4 adapters optimized as:

```
LINK SUPPORT
    BUFFERS 14 4202
```

or 14 buffers at 4202 bytes each, requires a transmit buffer in *protocol.ini* set to:

```
XMITBUFSIZE4208
```



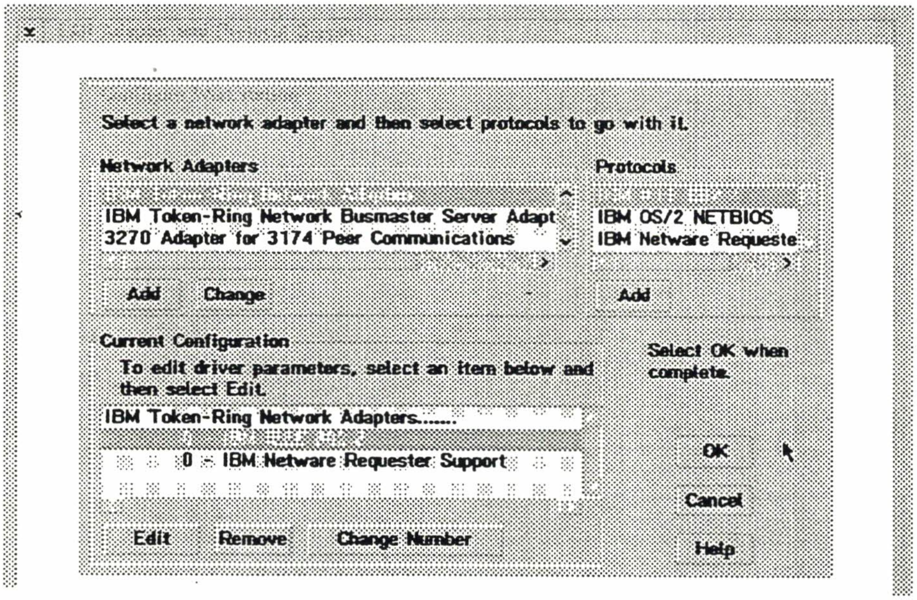
CAPTION - LANSUP enables Novell's ODI stack to sit on top of the IBM Extended Services NDIS-based architecture. Notice that the ODI stack is pushed way up. The resulting diagram defies almost all resemblance to the OSI model. The placing of the NetWare Requester on top of IBM communication interfaces was the original way of configuring Novell - IBM interoperability and dates back to the early days of the NetWare Requester on IBM's OS/2, Extended Edition 1.1.

which is six bytes larger than 4202. (The XMITBUFSIZE value must be a multiple of eight.) In other words, the *net.cfg*buffers cannot be larger or greater in number than whatever the *protocol.ini* specifies. The NDIS MAC layer driver (IBMTOK-nif [- underscore]), which was disabled in the *protocol.ini* for ODINSUP.SYS, must remain active with *lansup.sys*. Node address needs to be defined in the *net.cfg*even if the burned-in address is used.

In the *config.sys* file, path lines are essentially the same as they were in the *odinsup.sys*configuration. In addition, Protocol Manager and NDIS driver lines are similarly positioned to precede the NetWare Requester section as shown in the ODINSUP illustrations. In the NetWare Requester section, *lansup.sys* replaces TOKEN.SYS, and *odinsup.sys*is not present. *route.sys*will be needed on bridged Token-Rings as usual. A partial NetWare Requester section listing is as follows.

```
SET NWLANGUAGEENGLISH
DEVICEC:\NETWARE\LSL.SYS
RUNC:\NETWARE\DDAEMON.EXE
devicec:\netware\lansup.sys
DEVICEC:\NETWARE\ROUTE.SYS
DEVICEC:\NETWARE\IPX.SYS
DEVICEC:\NETWARE\SPX.SYS
```

IBM has a more direct approach, via the NTS/2 package, for placing ODI on top of NDIS. ODI2NDI-nif and ODI2NDI.SYS are respectively placed in the protocol.ini and config.sys files via the LAPS program. ODI2NDI is like ODINSUP in reverse. Rather than place NDIS on top of ODI, ODI is positioned above NDIS. ODI2NDI differs from LANSUP because LANSUP requires the 802.2 interface of LANDD.OS2 (and LANDLLDD.OS2) between it (LANSUP) and the lower level NDIS architecture. NDIS binding can occur before the NetWare Requester with LANSUP. In contrast, ODI2NDI always binds after the NetWare Requester section in the config.sys. The IBM LAPS program places the IBM directories, such as \IBMCOM, in the front rather than at the end of the PATH lines as illustrated in the Novell documentation.



Select a network adapter and then select protocols to go with it.

Network Adapters

IBM Token-Ring Network Busmaster Server Adapt
3270 Adapter for 3174 Peer Communications

Protocols

IBM OS/2 NETBIOS
IBM Netware Requeste

Current Configuration

To edit driver parameters, select an item below and then select Edit.

IBM Token-Ring Network Adapters.......
0 - IBM Netware Requester Support

Select OK when complete.

LANSUP CONFIGURATION LISTINGS

* * * PROTOCOL.INI * * *

```
[PROT_MAN]
DriverName  PROTMAN$

[IBMLXCFG]
IBMTOK_nif  IBMTOK.NIF
LANDD_nif  LANDD.NIF

[LANDD_nif]
DRIVERNAME  LANDD$
BINDINGS  IBMTOK_nif
ETHERAND_TYPE  "I"
SYSTEM_KEY  0X0
OPEN_OPTIONS  0X2000
TRACE  0X0
```

```
LINKS  41
MAX_SAPS  5
MAX_G_SAPS  0
USERS  4

[IBMTOK_nif]
DRIVERNAME  IBMTOK$
PRIMARY
MAXTRANSMITS  12
RECVBUFS  2
RECVBUFSIZE  256
XMITBUFS = 1
XMITBUFSIZE  4208
```

CAPTION - With LANSUP, the standard IBM NDIS drivers are required. *lansup.sys* and Novell ODI do not hook directly into NDIS. Instead, they reside above the IBM communications interfaces and drivers. Link support buffer size in the *net.cfg*needs to be 4202 bytes, or six bytes smaller than XMITBUFSIZE.

* * * CONFIG.SYS * * *

PATH LINES

```
LIBPATH.;C:\OS2\DLL;C:\MUGLIB\DLL;C:\OS2\MDOS;C:\CMLIB\DLL;
C:\;C:\OS2\APPS\DLL;C:\NETWARE;C:\IBMCOM\DLL;

SET PATHC:\OS2;C:\MUGLIB;C:\OS2\SYSTEM;C:\OS2\MDOS\WINOS2;
C:\CMLIB;C:\CMLIB\APPN;C:\OS2\INSTALL;C:\;C:\OS2\MDOS;
C:\OS2\APPS;C:\NETWARE;L:\OS2;P:\OS2;


SET DPATHC:\OS2;C:\MUGLIB\DLL;C:\CMLIB;C:\CMLIB\APPN;
C:\OS2\SYSTEM;C:\OS2\MDOS\WINOS2;C:\OS2\INSTALL;C:\;
C:\OS2\BITMAP;C:\OS2\MDOS;C:\OS2\APPS;C:\NETWARE;C:\IBMCOM;
L:\OS2;
```

CAPTION - PATH lines are similar to those shown for the ODINSUP configuration. The position of \IBMCOM directories relative to \NETWARE can be affected by the sequence of software installation. In addition, \NETWARE precedes L:\OS2 and P:\OS2 because NetWare OS/2 utilities are stored locally.

IBM COMMUNICATIONS LINES

```
DEVICEC:\IBMCOM\LANMSGDD.OS2 /I:C:\IBMCOM
DEVICEC:\IBMCOM\PROTMAN.OS2 /I:C:\IBMCOM
DEVICEC:\IBMCOM\PROTOCOL\LANDD.OS2
DEVICEC:\IBMCOM\PROTOCOL\LANDLLDD.OS2

DEVICEC:\CMLIB\ACSLDLAN.SYS
RUNC:\OS2\EPW.EXE
devicec:\ibmcom\macs\ibmtok.os2
RUNC:\IBMCOM\PROTOCOL\LANDLL.EXE
RUNC:\IBMCOM\PROTOCOL\NETBIND.EXE
RUNC:\IBMCOM\LANMSGEX.EXE
DEVICEC:\CMLIB\APPN\CMKFMDE.SYS
```

CAPTION - Protocol Manager (PROTMAN.OS2) loads prior to the 802.2 interface LAND drivers. IBMOK.OS2 is the IBM MAC layer driver for NDIS. The underlying configuration remains fairly standard from the IBM perspective.

CHANGES FOR ODI2NDI CONFIGURATION

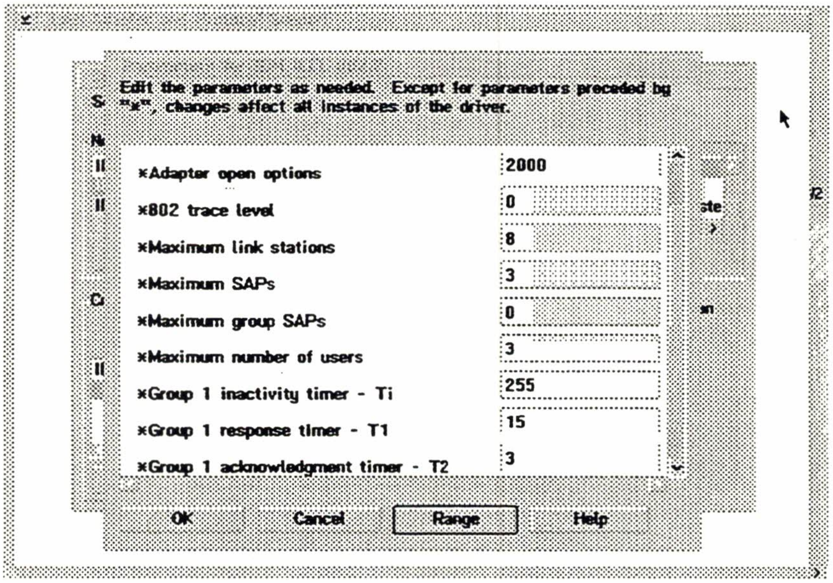
* * * PROTOCOL.INI * * *

```
[IBMLXCFG]
IBMTOK_nif  IBMTOK.nif
LANDD_nif  LANDD.nif
ODI2NDI_nif  ODI2NDI.nif

[ODI2NDI_nif]
DRIVERNAME  ODI2NDI$
BINDINGS  IBMTOK_nif
TOKEN_RING  "YES"
TOKEN_RING_SNAP  "YES"
ETHERNET_802.3  "NO"
ETHERNET_802.2  "NO"
ETHERNET_II  "NO"
ETHERNET_SNAP  "NO"
TRACE  0X0
```

CAPTION - The LAPS program makes an ODI2NDI entry into the IBMLXCFG section of the *protocol.ini* file and adds an ODI2NDI-nif [- UNDERSCORE] section which specifies frame types resembling those we saw in NET.CFG. The workstation is set up for Token-Ring.

* * * CONFIG.SYS * * *

```
rem Lines inserted before the NetWare Requester section:
DEVICEC:\IBMCOM\LANMSGDD.OS2 /I:C:\IBMCOM
DEVICEC:\IBMCOM\PROTMAN.OS2 /I:C:\IBMCOM

rem NetWare Requester section (partial view):
SET NWLANGUAGEENGLISH
DEVICEC:\NETWARE\LSL.SYS
RUNC:\NETWARE\DDAEMON.EXE
devicec:\ibmcom\protocol\odi2ndi.os2
DEVICEC:\NETWARE\ROUTE.SYS
DEVICEC:\NETWARE\IPX.SYS
DEVICEC:\NETWARE\SPX.SYS

rem After NetWare Requester section:
RUNC:\IBMCOM\PROTOCOL\NETBIND.EXE
RUNC:\IBMCOM\LANMSGEX.EXE
DEVICEC:\IBMCOM\PROTOCOL\LANDD.OS2
DEVICEC:\IBMCOM\PROTOCOL\LANDLLDD.OS2
devicec:\ibmcom\macs\ibmtok.os2
RUNC:\IBMCOM\PROTOCOL\LANDLL.EXE
```



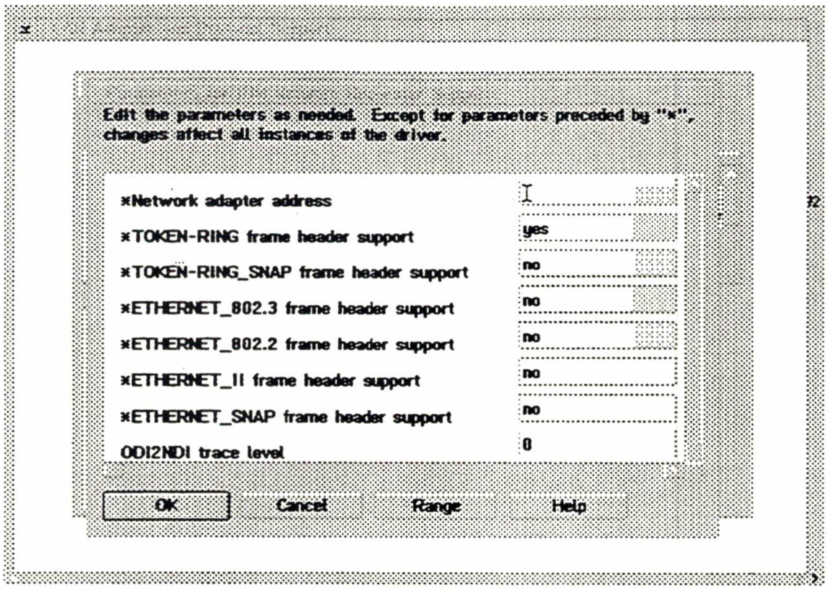
Edit the parameters as needed. Except for parameters preceded by "*", changes affect all instances of the driver.

*Adapter open options	2000
*802 trace level	0
*Maximum link stations	8
*Maximum SAPs	3
*Maximum group SAPs	0
*Maximum number of users	3
*Group 1 inactivity timer - Ti	255
*Group 1 response timer - T1	15
*Group 1 acknowledgment timer - T2	3

CAPTION - The LAPS program places most of its entries after the NetWare Requester section. ODI2NDI.OS2 becomes the driver to interface the NetWare Requester directly with NDIS, rather than through 802.2 (LANDD.OS2) such as in the *lansup.sys* configuration. ODI2NDI.SYS is inserted between LSL.SYS and *route.sys*where *token.sys*and *lansup.sys* were located in the configurations previously examined.

INTERWORKING ONWARD

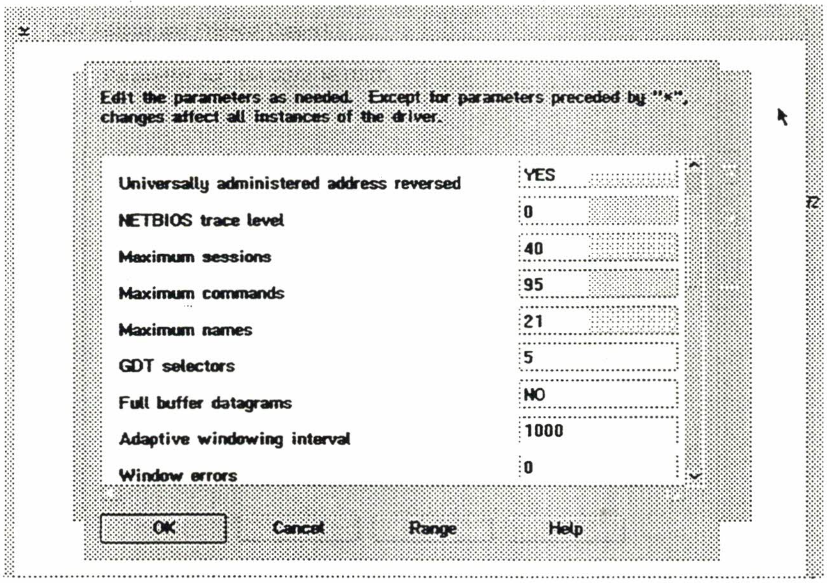
If LAN Services were also installed, additional lines would have been added to *config.sys* after the NetWare Requester section. Those lines load *netwska.200* (LAN Services file system), *netbios.os2* (IBM NetBIOS), and *lsdaemon.exe* (LAN Services daemon) followed by several IBMCOM executables: *landll.exe*, *netbind.exe* and then *lanmsgex.exe*.



Edit the parameters as needed. Except for parameters preceded by "*", changes affect all instances of the driver.

*Network adapter address	
*TOKEN-RING frame header support	yes
*TOKEN-RING_SNAP frame header support	no
*ETHERNET_802.3 frame header support	no
*ETHERNET_802.2 frame header support	no
*ETHERNET_II frame header support	no
*ETHERNET_SNAP frame header support	no
ODI2NDI trace level	0

The LAPs program can also be used to help configure the Novell NetWare Requester to run with the IBM ODI2NDI.SYS driver. Adapter frame types are specified to make *protocol.ini* file entries that will configure the adapter for the topology and network protocols to be initialized. Frame types follow the options for the NetWare Requester's *net.cfg*file, as discussed in the first article of this series.



Edit the parameters as needed. Except for parameters preceded by "*", changes affect all instances of the driver.

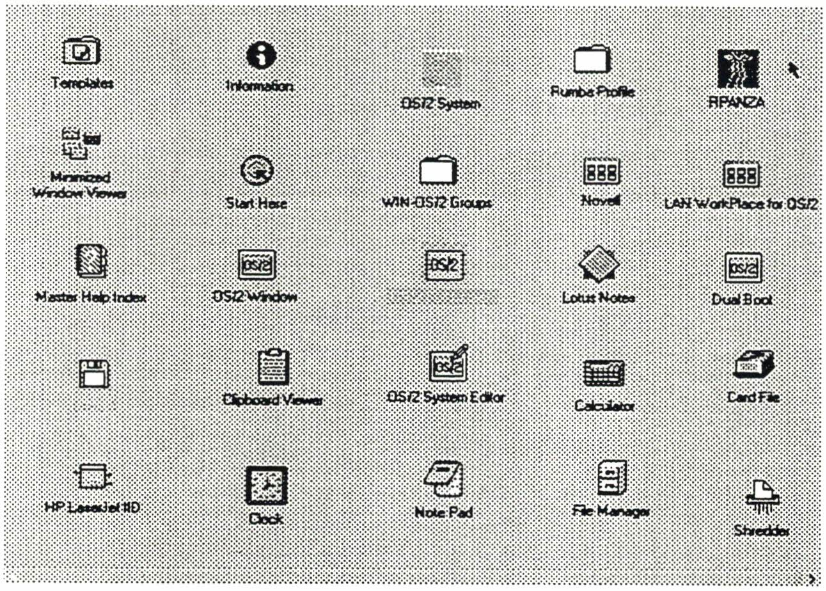
Universally administered address reversed	YES
NETBIOS trace level	0
Maximum sessions	40
Maximum commands	95
Maximum names	21
GDT selectors	5
Full buffer datagrams	NO
Adaptive windowing interval	1000
Window errors	0

OK
Cancel
Range
Help

IBM's OS/2 NetBIOS may be setup to support IBM LAN Requester or LAN Server. NetBIOS may be run concurrently with other protocols, such as SPX/IPX, TCP/IP, LLC, and Named Pipes. That means the OS/2 workstation can participate in native Novell, IBM, and Unix LAN functionality all at the same time. Token-Ring adapter architecture grants NetBIOS its own SAP apart from the 802.2 SAPs. In effect, all NetBIOS communication programs utilize the same SAP.

Beforehand, the three IBMCOM executables preceded the NetWare Requester in ODINSUP and LANSUP illustrations where LAN Services was not deployed. With LAN Services, the six files mentioned above must follow the NetWare Requester because the NetWare Requester file system (NWIFS.IFS) needs to be initialized prior to loading the LAN Services file system. The presence of LAN Services also means additions to the *protocol.ini* file and other configuration modifications to concurrently accomodate IBM NetBEUI and Novell NetBIOS.

What's next? The introduction of TCP/IP will open up new interoperabiity horizons beyond those we have so far discussed for the OS/2 platform. Which OS/2 TCP/IP package shall we choose? Novell's LAN WorkPlace or IBM's TCP/IP for OS/2? How does the choice impact the configuration of OS/2 network nodes and their capabilities across the enterprise? Stay tuned for adventures in OS/2 TCP/IP implementation.



Templates
Information
OS/2 System
Rumba Profile
RPANZA
Minimized Window Viewer
Start Here
WIN-OS/2 Groups
Novell
LAN WorkPlace for OS/2
Master Help Index
OS/2 Window
OS/2
Lotus Notes
Dual Boot
Clipboard Viewer
OS/2 System Editor
Calculator
Card File
HP LaserJet IID
Clock
Note Pad
File Manager
Shredder

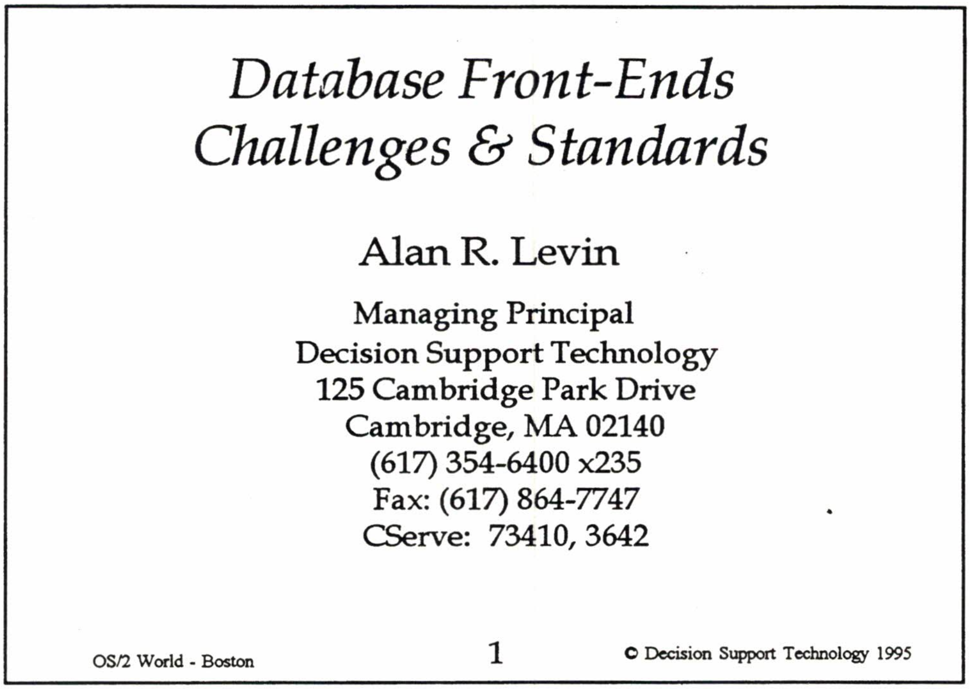
Various interoperability configurations make many things possible on the desktop. Novell's Open Data Link Interface (ODI) architecture was designed to be structurally accomodative. Multiprotocol communication adds functional flexibility for interworking across diverse enterprisewide networks.

this page intentionally left blank

SYS 0506

Database Front Ends: Challenges, Standards and Rules

Alan Levin
Decisions Support Technology
Cambridge, Mass.

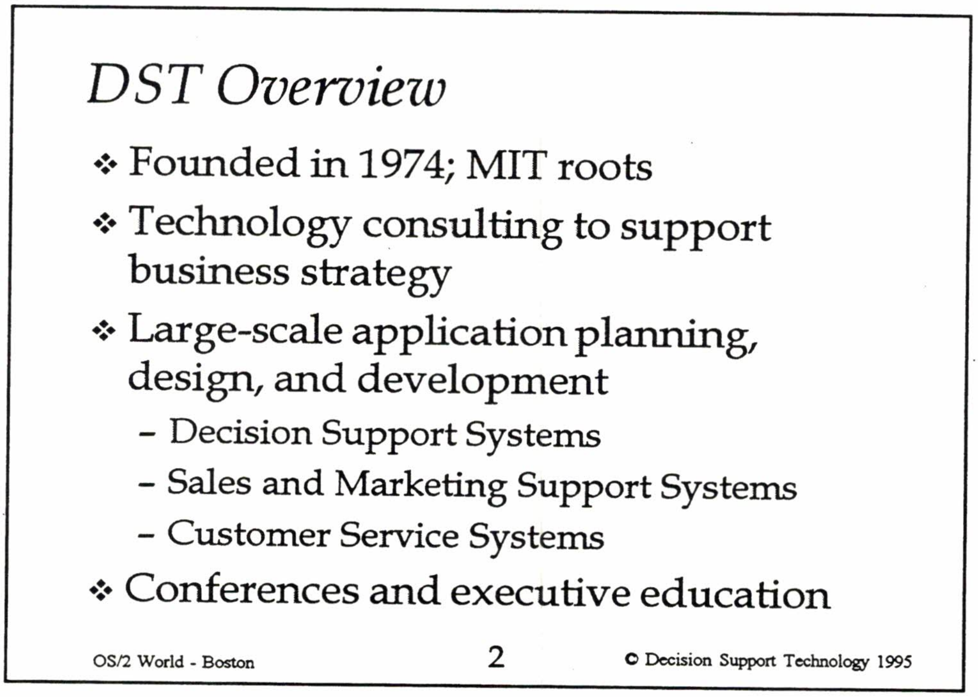


Database Front-Ends Challenges & Standards

Alan R. Levin

Managing Principal
Decision Support Technology
125 Cambridge Park Drive
Cambridge, MA 02140
(617) 354-6400 x235
Fax: (617) 864-7747
CServe: 73410, 3642

© Decision Support Technology 1995



DST Overview

- Founded in 1974; MIT roots
- Technology consulting to support business strategy
- Large-scale application planning, design, and development
  - Decision Support Systems
  - Sales and Marketing Support Systems
  - Customer Service Systems
- Conferences and executive education

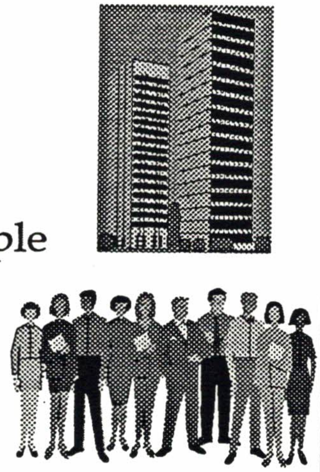
© Decision Support Technology 1995

Agenda

- ❖ Overview of Issues
- ❖ Overview of Choices
- ❖ The Roadmap
- ❖ Case Studies
- ❖ Keys to Success
- ❖ Summary

© Decision Support Technology 1995

Database Applications



- ❖ Good for the company
- ❖ Usable by "ordinary" people

© Decision Support Technology 1995

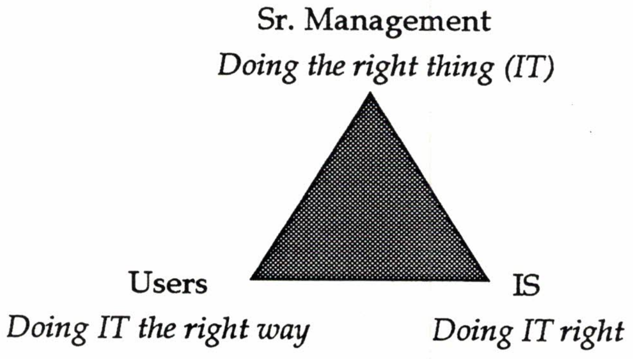
Role of Database Applications

- Move from qualitative to analytical scenarios
- Identify new strategic directions
- Fine-tune tactical plans
- Systematize performance measurement
- Improve organizational communication

Facilitate Change

© Decision Support Technology 1995

Successful Execution



Sr. Management

Doing the right thing (IT)

Users

Doing IT the right way

IS

Doing IT right

© Decision Support Technology 1995

Doing the Right Thing (IT)

- ❖ Expected return on investment
- ❖ Leveragability for other projects
- ❖ Raising competitive barriers
- ❖ Results: Cheaper, Faster, Better

OS/2 World - Boston

© Decision Support Technology 1995

Doing IT Right

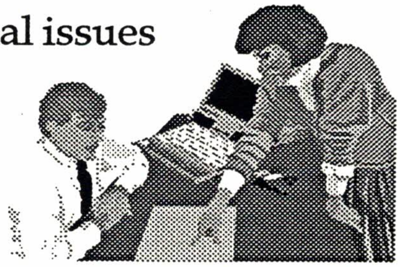
- ❖ Leverage existing resources and technologies
- ❖ Assess technical tradeoffs
- ❖ Properly manage time and money
- ❖ Implement open systems for growth and change over time

OS/2 World - Boston

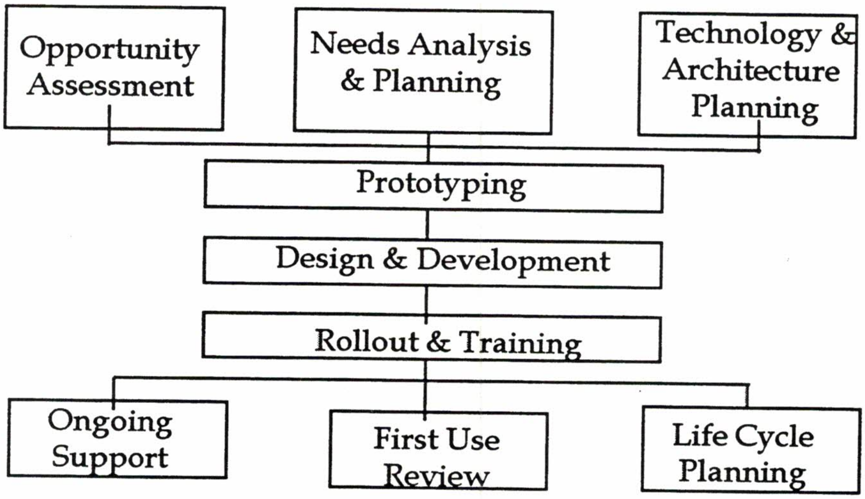
© Decision Support Technology 1995

Doing IT the Right Way

- Continuous user feedback
- User training
- Adapt to organizational readiness
- Address organizational issues
- Meet user needs



DST Methodology



Opportunity Assessment

Needs Analysis & Planning

Technology & Architecture Planning

Prototyping

Design & Development

Rollout & Training

Ongoing Support

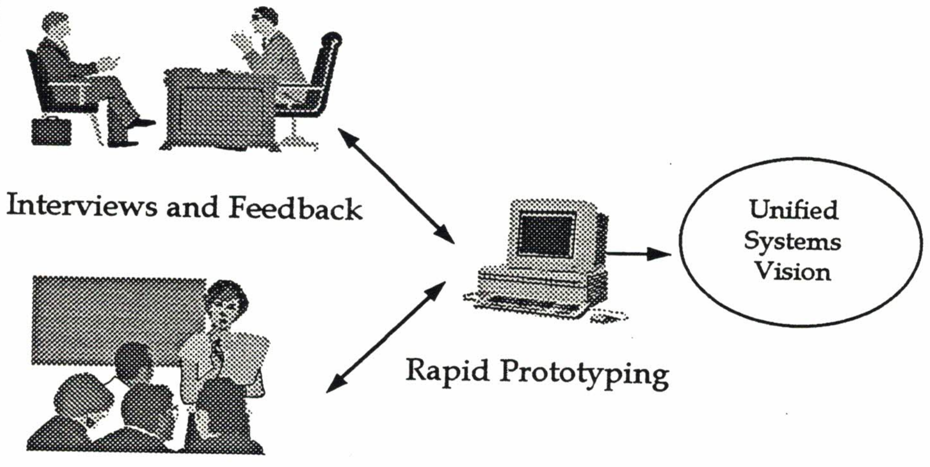
First Use Review

Life Cycle Planning

Approach

- Quickly demonstrate strategic value and tactical benefits
- Make end users an integral part of the development team
- Proactively address organizational change
- Use rapid prototyping techniques

Input and Feedback



Interviews and Feedback

Rapid Prototyping

Unified Systems Vision

Project Team Focus

System Design and Architecture

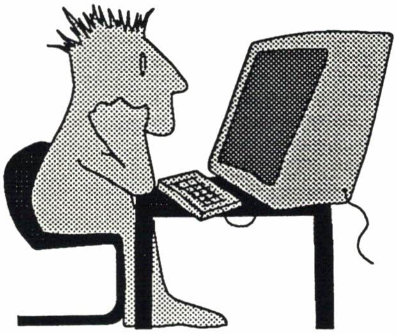
- ❖ Making the back-end work
- ❖ Tools Integration -- issues and answers
- ❖ Consistency in data definitions
- ❖ Technical consensus on approach

Professional Business Systems Are Different

Operational Business Systems	**Professional Business Systems**
• Meet corporate needs • Structured problems • Focus on data • Improve efficiency • Lower cost	• Meet user needs • Unstructured problems • Focus on results • Improve effectiveness • Increase revenue

Challenges & Issues

- User Function
- User Capabilities
- Business Goals
- Current Technology
- New Technology
- Onsite vs. Mobile
- Functional Rqmts

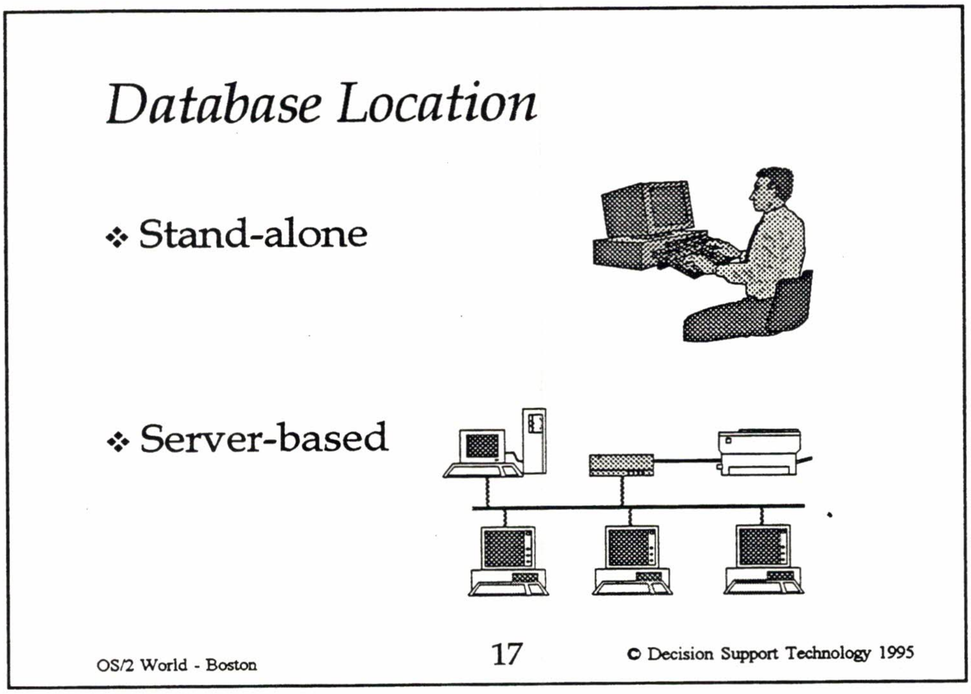


© Decision Support Technology 1995

The Roadmap

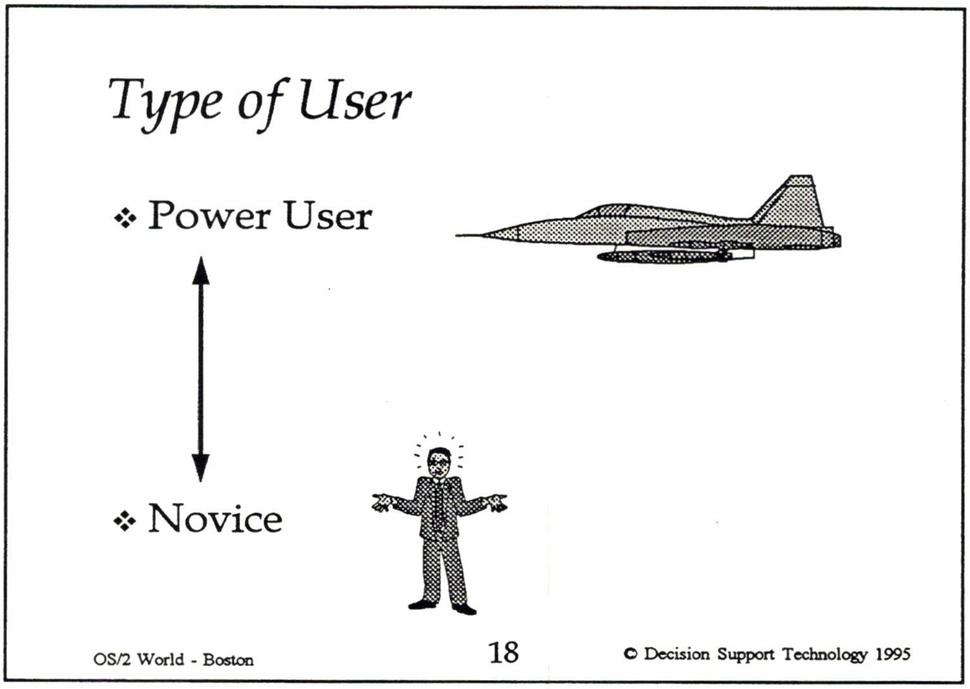
- Determine Functional Requirements
- Determine Technical Requirements
- Evaluate User Capabilities
- Review alternatives
- Select best fit product and approach
- Prototype
- Full system development

© Decision Support Technology 1995



Database Location

- Stand-alone
- Server-based



Type of User

- Power User
- Novice

Functional Requirements

- Ease of Use
- Adhoc query and analysis
- Reporting
- Statistical analysis and modeling
- Customizable
- Fixed vs. Flexible Format

Technical Requirements

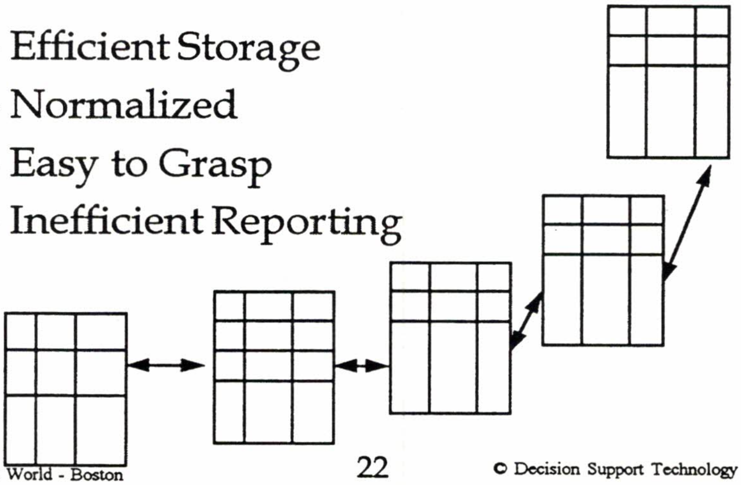
- Portability
- Maintainability
- Adaptability
- Meets corporate standards & guidelines
- Development speed & capabilities

The Database Model

- ❖ Relational
- ❖ Multi-Dimensional
- ❖ Flat-file

The Relational Database

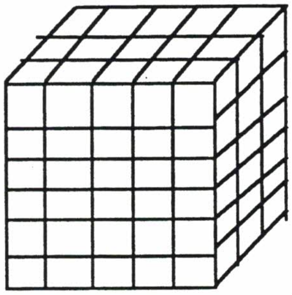
- ❖ Efficient Storage
- ❖ Normalized
- ❖ Easy to Grasp
- ❖ Inefficient Reporting



Multidimensional Database

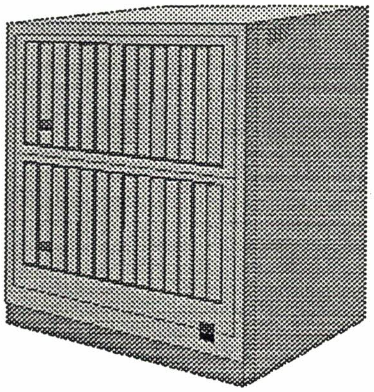
- Efficient Access
- Handles time easily
- Flexible reporting

- Inefficient Storage



Flat-File Database

- Inefficient Storage
- No adhoc Reporting
- Easy to maintain
- Transaction Efficient



Application Partitioning

- When to use the client
- When to use the server

OS/2 World - Boston

© Decision Support Technology 1995

Client Database for Processing

- Distributed computing architecture
- Mobile users
- Remote Processing / Central Collection
- Limited database transactions

OS/2 World - Boston

© Decision Support Technology 1995

Server Database Processing

- Shared, real-time data
- Significant number of transactions
- Centralized distribution of information

Issues and Problems

- Features
- Technology Platform
- Price
- Performance
- Vendor Stability

Front-End Alternatives

- C++
- Visual Basic
- Database tools (e.g. Oracle Forms)
- Powerbuilder, SQL Windows, Objectview
- SQL Plus, Business Objects, GQL
- Other (e.g. RemoteWare Forms)

Choosing a Front-End: Tradeoffs

- Type of user
- Type of application
- Product vs. Customized Application
- Processing speed requirements
- Target hardware platform
- Programmability / Maintainability
- Implementation Timeframe

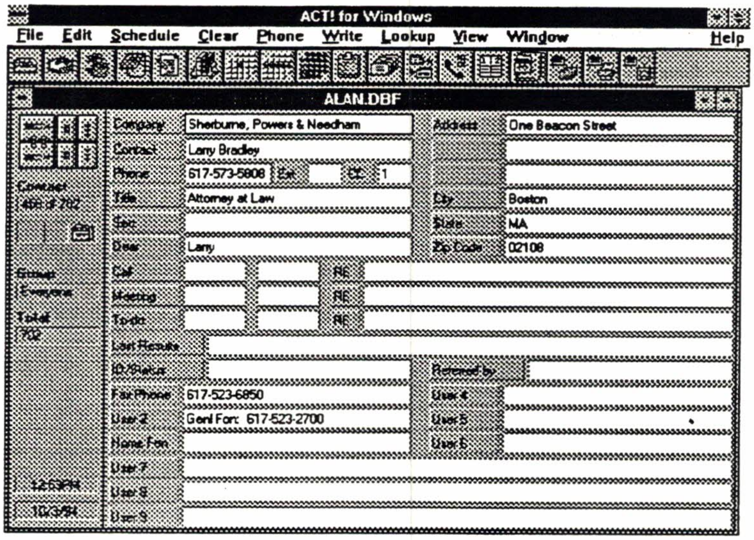
The Standalone Solution

- ❖ No integration requirements
- ❖ Cross-package tie-ins done directly
- ❖ Make use of operating system capabilities e.g. cut-and-paste

Case 1
Overview

- ❖ Contact Management
- ❖ Users: Salespeople
- ❖ Need: Stand-alone Product
- ❖ Single User Application
- ❖ Database Front End: C++
- ❖ Graphical User Interface

Case 1


ACT! for Windows
File Edit Schedule Clear Phone Write Lookup View Window Help
ALAN.DBF
Contact
468 of 702
Group
Everyone
Total
702
10/3/94

Company: Sherburne, Powers & Needham

Contact: Larry Bradley

Phone: 617-573-5808 Ext: CC: 1

Title: Attorney at Law

Sec:

Dear: Larry

Last Results:

ID/Status:

Fax Phone: 617-523-6850

User 2: Genl Fax: 617-523-2700

Home Fax:

Address: One Beacon Street

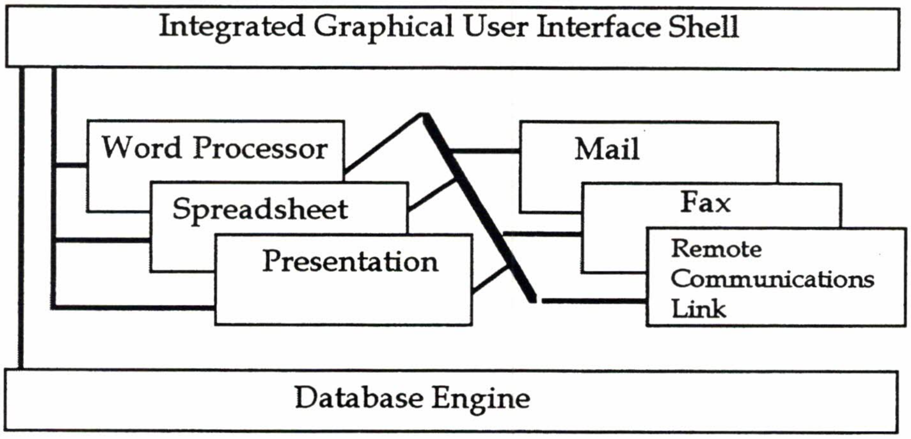
City: Boston

State: MA

Zip Code: 02108

Referred by:

Integrated Software Environment



Integrated Graphical User Interface Shell

Word Processor

Spreadsheet

Presentation

Mail

Fax

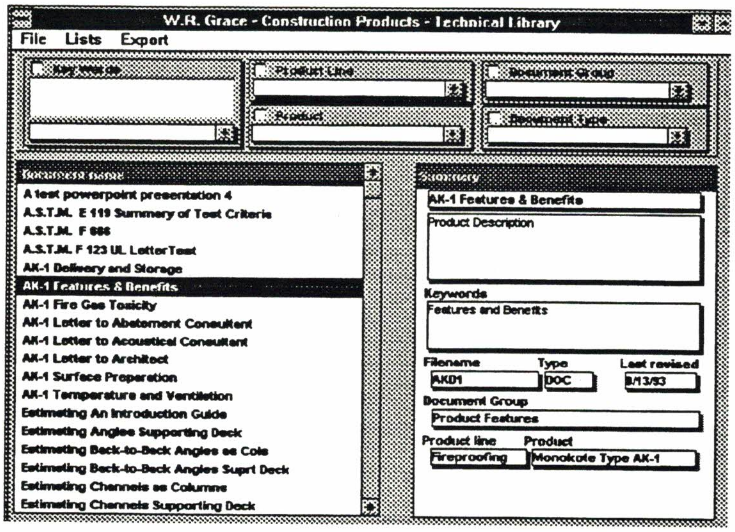
Remote Communications Link

Database Engine

Case 2
Overview

- ❖ Technical Library
- ❖ Users: Salespeople
- ❖ Application: Sales Support
- ❖ Need: Integrated Software Environment
- ❖ Same database on server & notebook
- ❖ Powerbuilder - Watcom SQL - MS Office

Case 2


W.R. Grace - Construction Products - Technical Library
File
Lists
Export
☐ Key Words
☐ Product Line
☐ Document Group
☐ Product
☐ Document Type
Document name
Summary

A test powerpoint presentation 4
A.S.T.M. E 119 Summary of Test Criteria
A.S.T.M. F 686
A.S.T.M. F 123 UL LetterTest
AK-1 Delivery and Storage
AK-1 Features & Benefits
AK-1 Fire Gas Toxicity
AK-1 Letter to Abatement Consultant
AK-1 Letter to Acoustical Consultant
AK-1 Letter to Architect
AK-1 Surface Preparation
AK-1 Temperature and Ventilation
Estimating An Introduction Guide
Estimating Angles Supporting Deck
Estimating Back-to-Back Angles as Cols
Estimating Back-to-Back Angles Suprt Deck
Estimating Channels as Columns
Estimating Channels Supporting Deck

AK-1 Features & Benefits

Product Description

Keywords: Features and Benefits

Filename: AK01
Type: DOC
Last revised: 8/13/93

Document Group: Product Features

Product line: Fireproofing
Product: Monokote Type AK-1

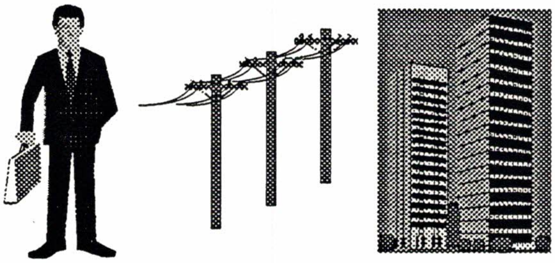
Portability Across Servers

- ❖ Notebook
- ❖ Desktop
- ❖ Communication Server
- ❖ Execution Server
- ❖ File Server
- ❖ Database Server

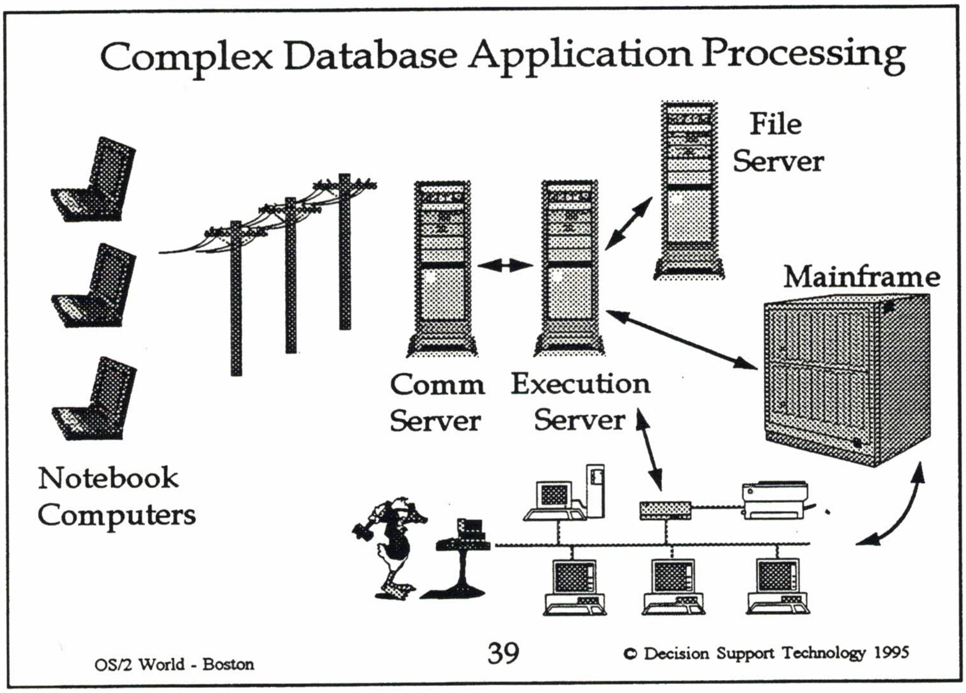
© Decision Support Technology 1995

Case 3

- ❖ Mobile Computing application
  Sales Force Automation



© Decision Support Technology 1995



Complex Database Application Processing

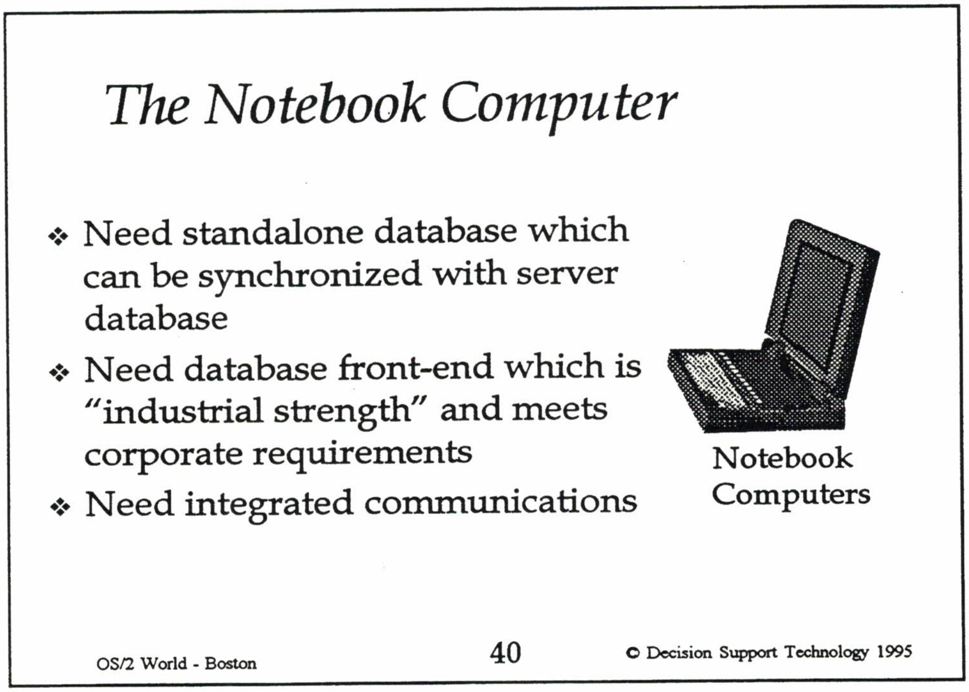
File Server

Mainframe

Comm Server

Execution Server

Notebook Computers



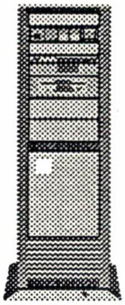
The Notebook Computer

- Need standalone database which can be synchronized with server database
- Need database front-end which is "industrial strength" and meets corporate requirements
- Need integrated communications

Notebook Computers

The Communications Server

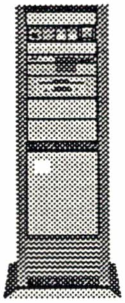
- ❖ Used to management transfer of files and data between remote and central site
- ❖ No database processing
- ❖ Pass-through device



Communications Server

The Database Server

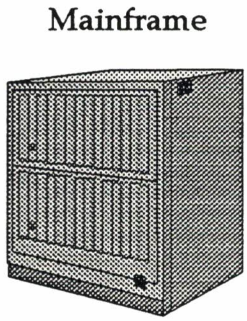
- ❖ Centralized storage of data from multiple sites or processors
- ❖ Will have database processing
- ❖ May require database front-end or may not depending upon function and accessibility by users



Database Server

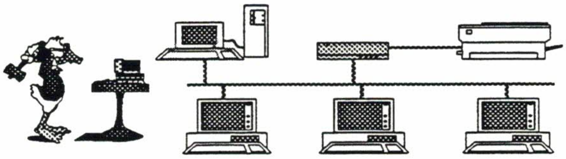
The Mainframe Computer

- Large data warehouse
- Large volume data processing
- Need to limit access to transaction based data for security and responsiveness
- Front-End applications usually standard reports or direct "non-friendly" access



The Local Area Network

- Facilitates linkages between users
- Allows for simultaneous access and updating
- Mixed technology networks common
- May need multi-platform front-end development tool to minimize time and costs

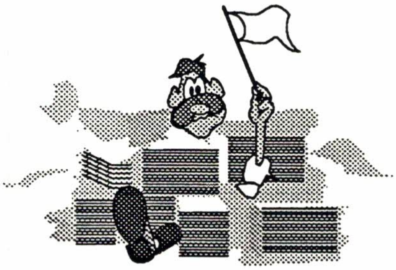


Case 3
Results

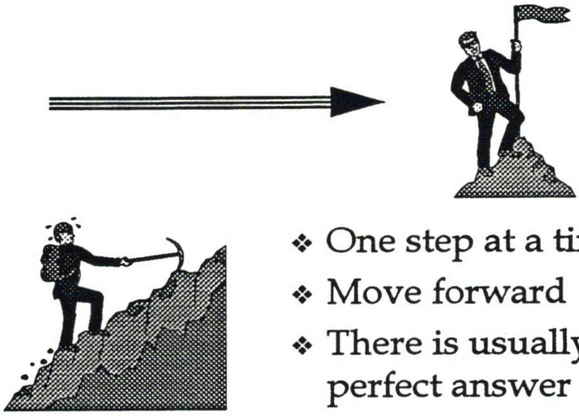
- ❖ Integration of data across divisions
- ❖ Increased sales productivity
- ❖ Increased volume, dollars & profits
- ❖ Reduced demands on customer service
- ❖ Better customer satisfaction

Too Much to handle?

- ❖ Too many choices?
- ❖ Too many obstacles?
- ❖ Organizational Issues?
- ❖ Technical Issues?
- ❖ No easy answer?



The Journey to Success



- One step at a time
- Move forward
- There is usually not a perfect answer

Keys to Success

- Meet management requirements
- Meet corporate IS linkage requirements
- Minimal training -- keep it simple
- Match the "way they think and work"
- Rollout in less than 6 months
- Manage expectations; avoid roadblocks

CS 0605

CICS OS/2: 32-bit Client/Server Computing

Jim McNair
IBM

CICS

Application
Server

Scalable Mission Critical Client/Server

CICS
Application Server

Copyright 1994 IBM Corporation

Trademarks

The following terms used in this presentation are trademarks or (R) registered trademarks of the IBM Corporation in the United States and/or in other countries:

IBM(R)
CICS
CICS/ESA
CICS/MVS
CICS/VSE
CICS OS/2
CICS/400
CICS/6000
MVS
VSE
AIX(R)
OS/2(R)
OS/400(R)
LANDP
PS/2(R)
ES/9000
RISC System/6000
AS/400
DB2
SQL/DS

The following terms are trademarks of other companies or institutions:

UNIX	UNIX System Laboratories, Inc (Registered Trademark)
OSF	Open Systems Foundation, Inc
DCE	Open Software Foundation, Inc
Oracle	Oracle Corporation
Informix	Informix Software Inc
Ingres	Ingres Corporation
Sybase	Sybase Inc

CICS
Application Server

Copyright 1994 IBM Corporation

~~CS 0605 CICS OS/2~~
McNair

Client/Server Today

Benefits
- ★ User Productivity
- ★ Rapid Application Development
- ★ Graphical Interfaces
- ★ End User Control
- ★ Cost Savings

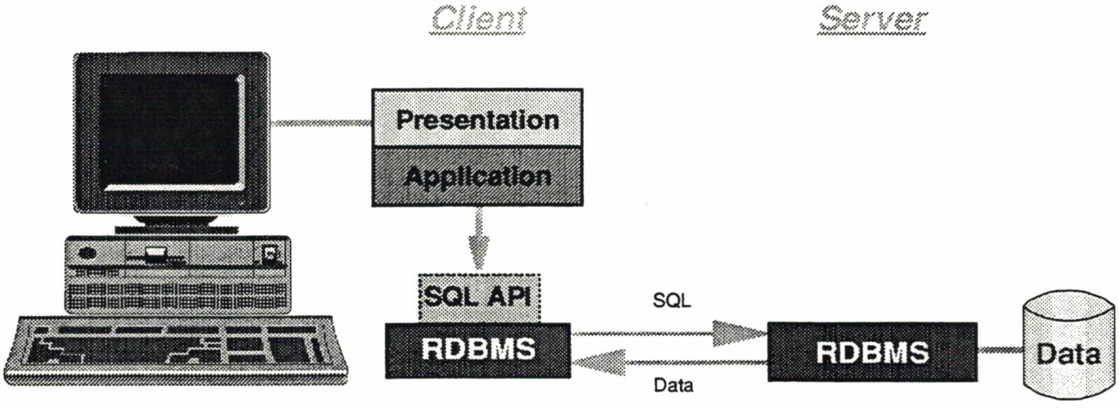
Needs
- ★ Mission Critical Infrastructure
- ★ Support Large Scale Systems

Generally Departmental, Data Oriented

CICS
Application Server

Copyright 1994 IBM Corporation

Data Server (2-Tier)


Client
Server
Presentation
Application
SQL API
RDBMS
SQL
Data
RDBMS
Data

- Remote Data Retrieved by Requests Through Local Data Management
- Business Logic on Client
- Network I/O for Each Data Request (SQL Statement)
- Focus on Data

CICS
Application Server

Copyright 1994 IBM Corporation

Challenges of 2-Tier Approach

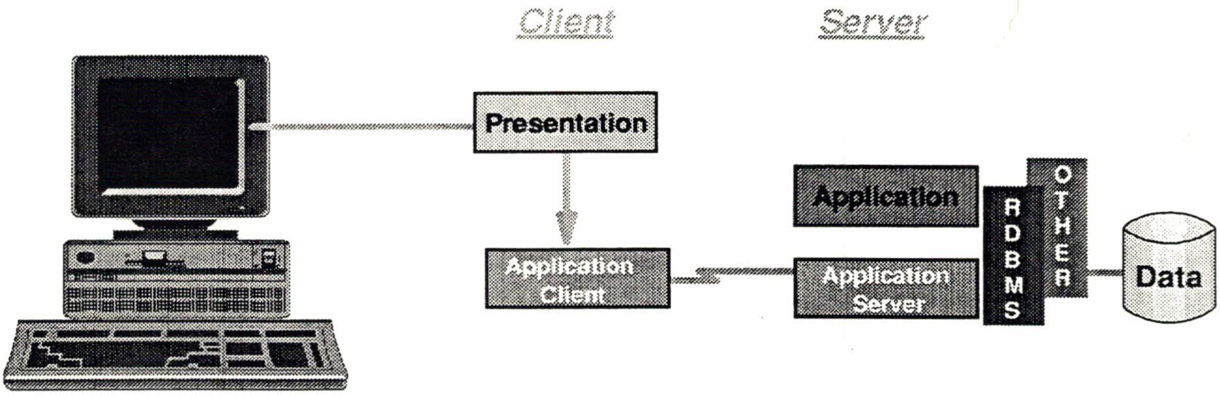
- ✓ Scalable Performance
  - ★ Network Traffic
  - ★ Data Locking
- ✓ Control/Management
- ✓ Security
- ✓ End to End Integrity

CICS

Application Server

Copyright 1994 IBM Corporation

Application Server (3-Tier)


Client
Server
Presentation
Application Client
Application
Application Server
RDBMS
OTHER
Data

- Business Logic in Server Applications
- Logical Business Request Sent to Server
- One Network I/O per Logical Request
- Focus on Application

CICS

Application Server

Copyright 1994 IBM Corporation

Application Server Advantages

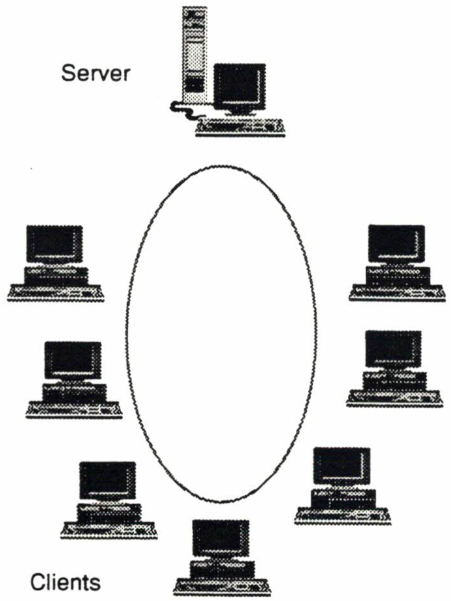
- ✓ Scalable Performance
- ✓ Control of Business Processes
- ✓ Security
- ✓ Heterogeneous Data
- ✓ Client Costs

CICS
Application
Server

Copyright 1994 IBM Corporation

Scalable Performance

- ✓ Minimize Network Traffic
  - ★ Logical vs Data Request
- ✓ Data Locking
  - ★ Conversational Locks
- ✓ Interpretive

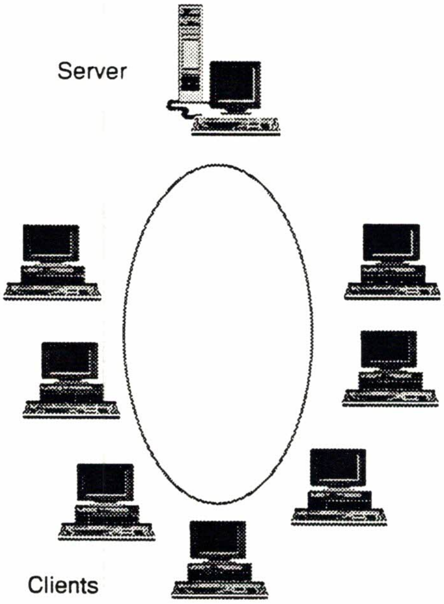


CICS
Application
Server

Copyright 1994 IBM Corporation

Application Control & Management

- ✓ Client Environment
- ✓ Software Distribution
  - ★ Off-Shift Access
- ✓ Version Control
- ✓ Number of Clients vs Servers



CICS Application Server

Copyright 1994 IBM Corporation

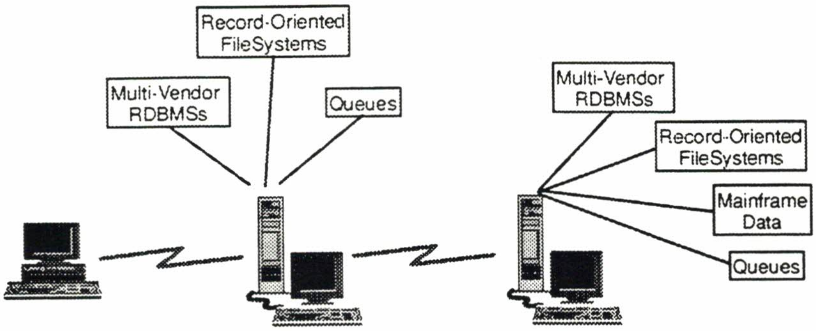
Application Security

- ✓ No Direct Access to Data
- ✓ Limit How Data is Used
  - ★ Rather than just which data
- ✓ Protect Against
  - ★ Malicious Intent
  - ★ Inadvertent Destruction
- ✓ Enforce Business Rules
- ✓ Additional Criteria
  - ★ Specific Locations, etc

CICS Application Server

Copyright 1994 IBM Corporation

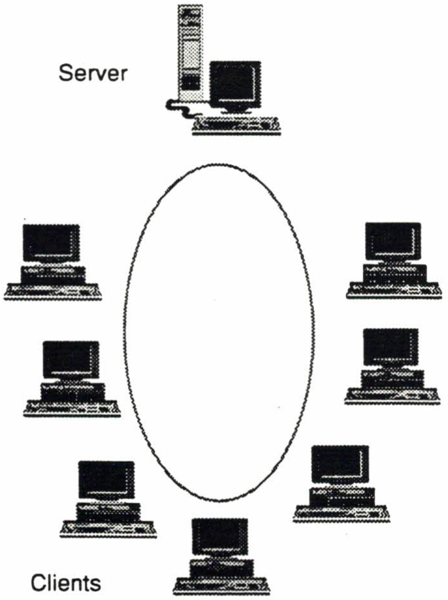
Heterogeneous Data Access


Record-Oriented FileSystems
Multi-Vendor RDBMSs
Queues
Multi-Vendor RDBMSs
Record-Oriented FileSystems
Mainframe Data
Queues

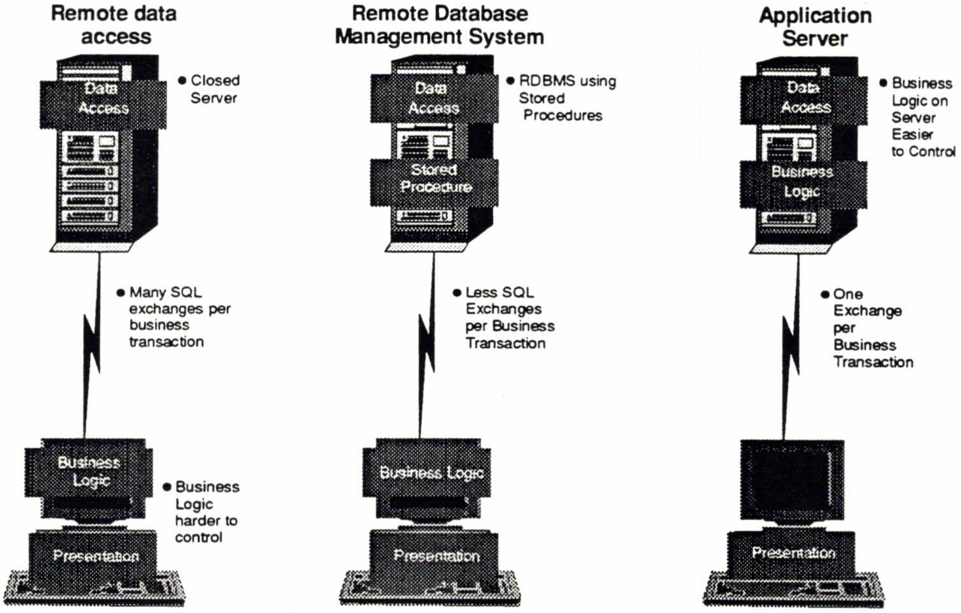
- ✓ Need to Integrate Many Types of Data
- ✓ Data on Multiple Platforms
- ✓ Integration must include Data Integrity

Client Costs

- ✓ Many More Clients
  - ★ Hardware/Software
  - ★ Support
- ✓ Initial Size and Growth


Server
Clients

Data Oriented vs Application Oriented Servers


Data Access
Data Access
Stored Procedure
Data Access
Business Logic
Business Logic
Presentation
Business Logic
Presentation
Presentation

Remote data access

- Closed Server
- Many SQL exchanges per business transaction
- Business Logic harder to control

Remote Database Management System

- RDBMS using Stored Procedures
- Less SQL Exchanges per Business Transaction

Application Server

- Business Logic on Server Easier to Control
- One Exchange per Business Transaction

CICS
Application Server

Copyright 1994 IBM Corporation

CICNF140

A New Application Scenario

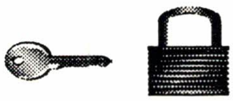
Developing New Client/Server Application

Problems Encountered

Listener Function

Preloaded Applications

Multi-Tasking Monitor



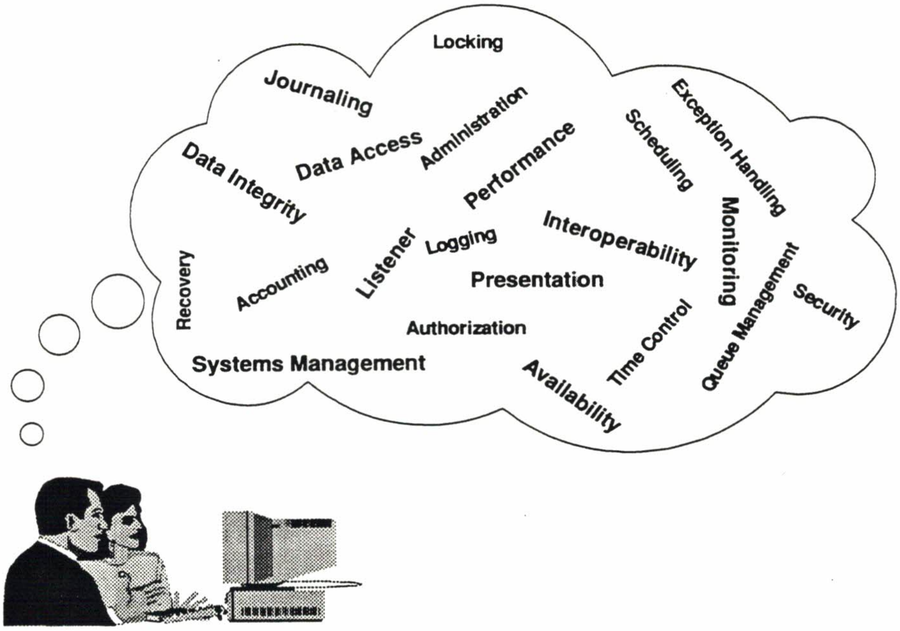
Security

CICS
Application Server

Copyright 1994 IBM Corporation

CICNF100

System Issues vs. Application

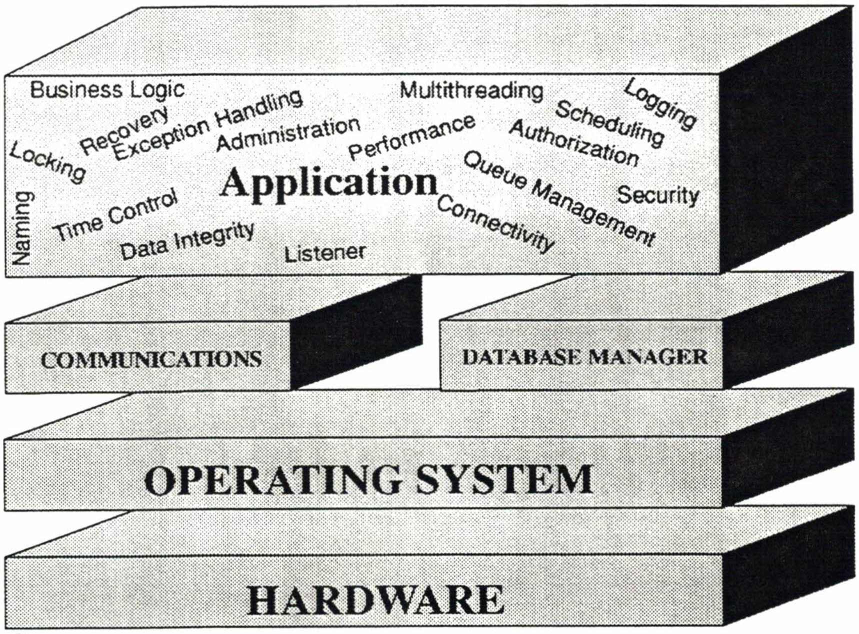


Locking
Journaling
Data Access
Administration
Exception Handling
Data Integrity
Performance
Scheduling
Interoperability
Monitoring
Logging
Listener
Presentation
Accounting
Recovery
Security
Queue Management
Authorization
Time Control
Systems Management
Availability

CICS Application Server

Copyright 1994 IBM Corporation

Business Application Requirements



Business Logic
Multithreading
Logging
Recovery
Exception Handling
Scheduling
Administration
Authorization
Locking
Performance
Application
Queue Management
Security
Naming
Time Control
Connectivity
Data Integrity
Listener

COMMUNICATIONS
DATABASE MANAGER

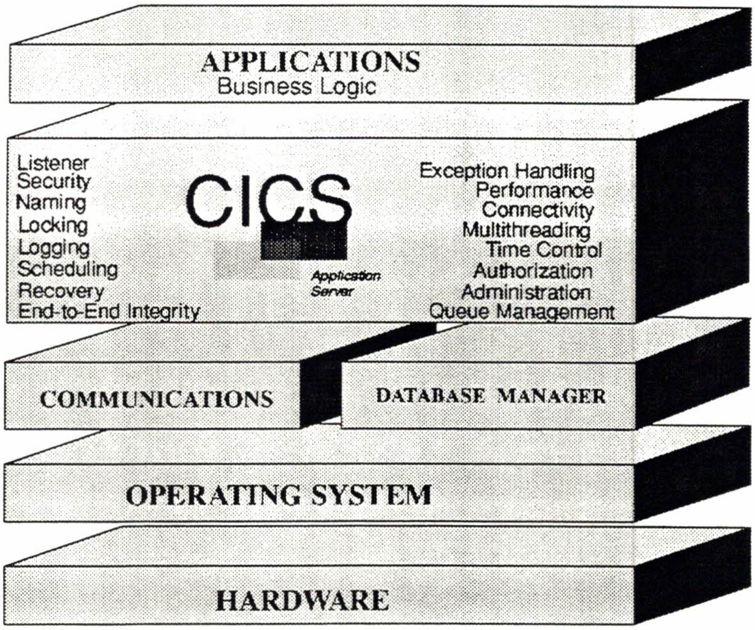
OPERATING SYSTEM

HARDWARE

CICS Application Server

Copyright 1994 IBM Corporation

Application Server Functions



APPLICATIONS
Business Logic

CICS
Application Server

Listener
Security
Naming
Locking
Logging
Scheduling
Recovery
End-to-End Integrity

Exception Handling
Performance
Connectivity
Multithreading
Time Control
Authorization
Administration
Queue Management

COMMUNICATIONS

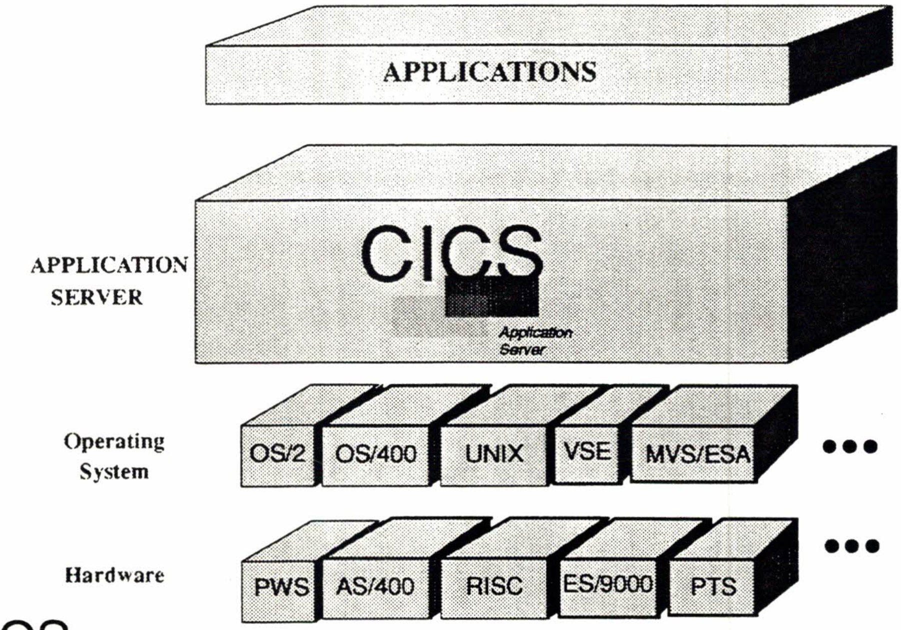
DATABASE MANAGER

OPERATING SYSTEM

HARDWARE

Copyright 1994 IBM Corporation

CICS Is Everywhere You Need It



APPLICATIONS

APPLICATION SERVER

CICS
Application Server

Operating System

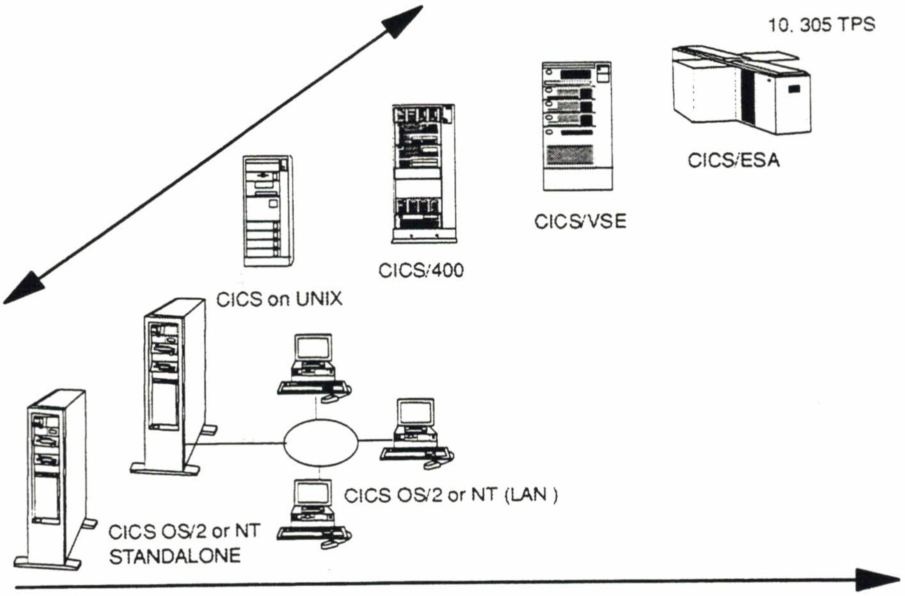
OS/2 | OS/400 | UNIX | VSE | MVS/ESA ...

Hardware

PWS | AS/400 | RISC | ES/9000 | PTS ...

Copyright 1994 IBM Corporation

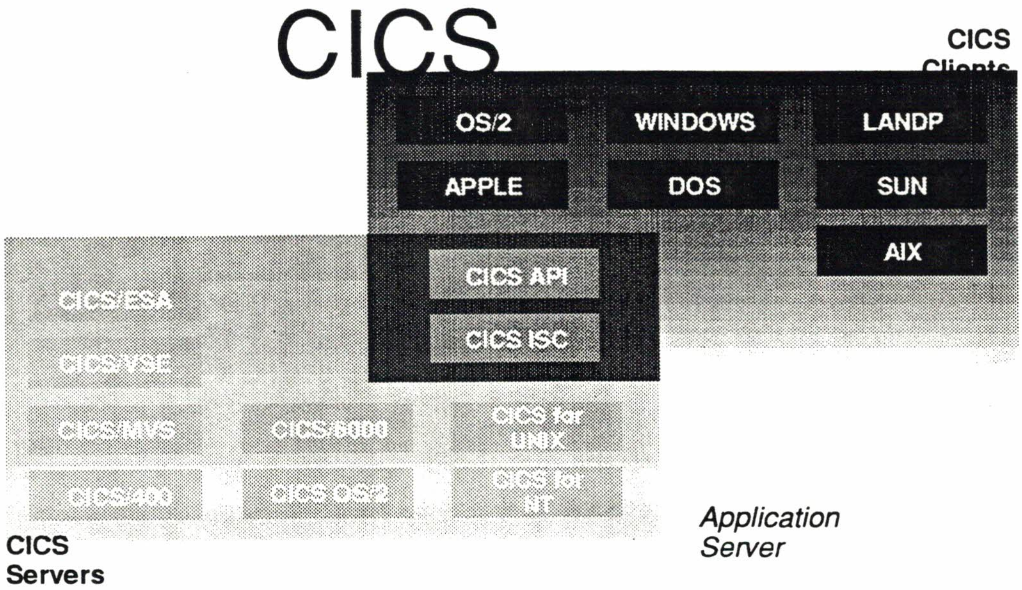
CICS Grows With Your Business


10. 305 TPS
CICS/ESA
CICS/VSE
CICS/400
CICS on UNIX
CICS OS/2 or NT (LAN)
CICS OS/2 or NT
STANDALONE

CICS
Application Server

Copyright 1994 IBM Corporation

Certainly Is Client Server



CICS

CICS Clients

OS/2
WINDOWS
LANDP
APPLE
DOS
SUN
AIX

CICS API
CICS ISC

CICS/ESA
CICS/VSE
CICS/MVS
CICS/6000
CICS for UNIX
CICS/400
CICS OS/2
CICS for NT

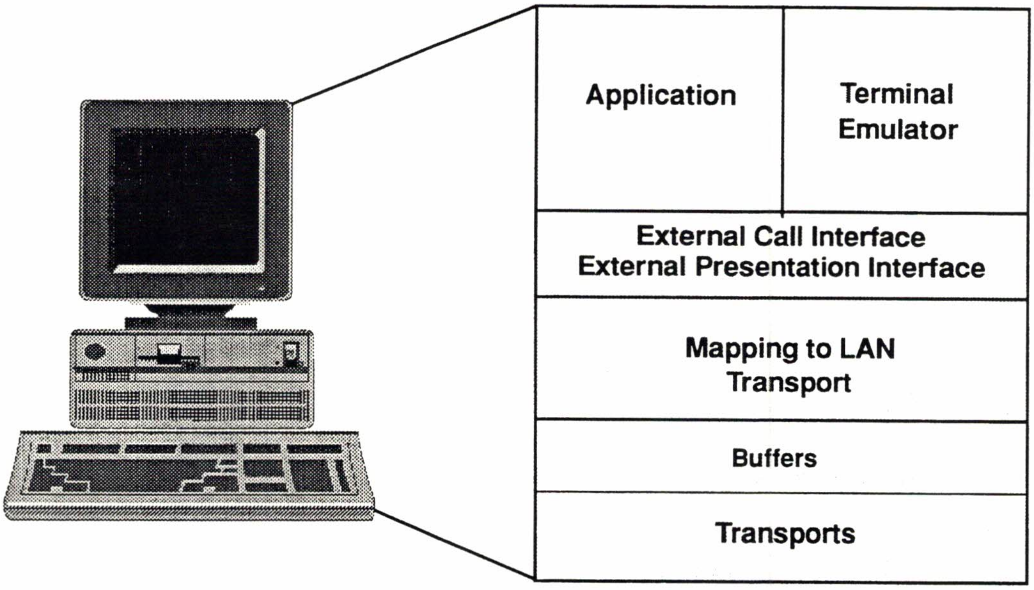
Application Server

CICS Servers

CICS
Application Server

Copyright 1994 IBM Corporation

CICS Client Architecture (Skinny Clients)



Application

Terminal Emulator

External Call Interface
External Presentation Interface

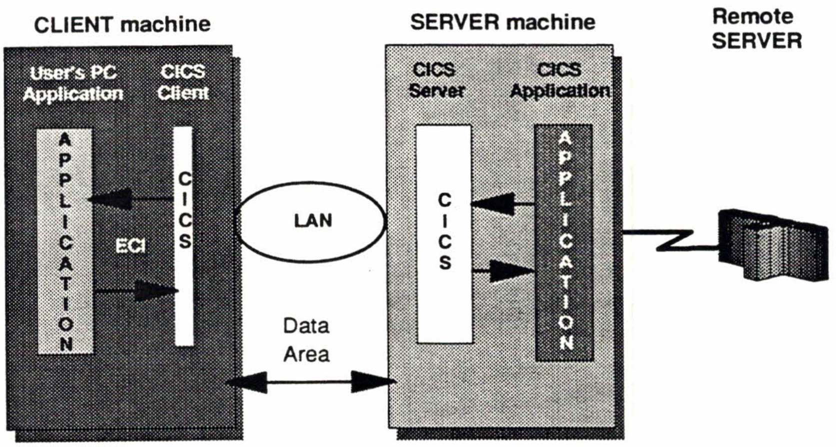
Mapping to LAN Transport

Buffers

Transports

Copyright 1994 IBM Corporation

External Call Interface (ECI)



CLIENT machine

SERVER machine

Remote SERVER

User's PC Application

CICS Client

APPLICATION

CICS

ECI

LAN

Data Area

CICS Server

CICS Application

CICS

APPLICATION

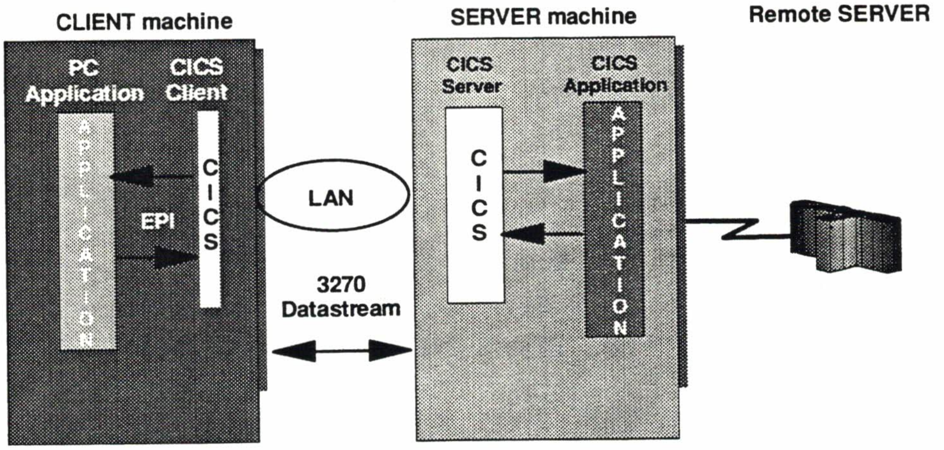
A method for a non-CICS Application to call a CICS Program as a subroutine

CICS application could be on server or remote server

Call / Parameter Driven

Copyright 1994 IBM Corporation

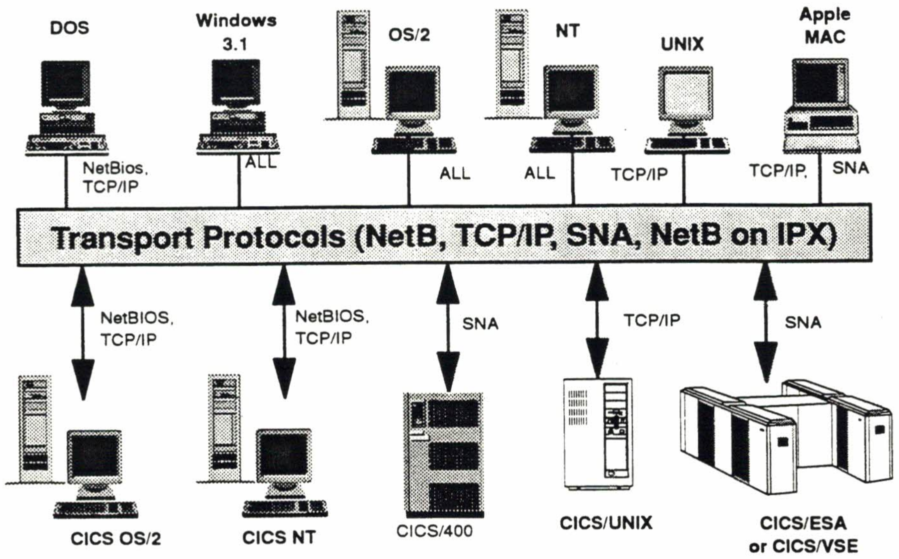
External Presentation Interface (EPI)


CLIENT machine
SERVER machine
Remote SERVER
PC Application
CICS Client
APPLICATION
CICS
EPI
LAN
3270 Datastream
CICS Server
CICS Application
CICS
APPLICATION

- A method for a PC program to pretend to be a 3270 and use a 3270 oriented CICS application
- Allows a GUI front end to be added to an EXISTING CICS application
  - without changing the CICS application
- CICS application request could be forwarded to remote server
- Data Stream or Event Driven
- Host can initiate applications for connected clients

Copyright 1994 IBM Corporation

CICS Client to Server Protocols


DOS
Windows 3.1
OS/2
NT
UNIX
Apple MAC
NetBios, TCP/IP
ALL
ALL
ALL
TCP/IP
TCP/IP,
SNA
Transport Protocols (NetB, TCP/IP, SNA, NetB on IPX)
NetBIOS, TCP/IP
NetBIOS, TCP/IP
SNA
TCP/IP
SNA
CICS OS/2
CICS NT
CICS/400
CICS/UNIX
CICS/ESA or CICS/VSE

Any Client, Any Protocol, Any Server

Copyright 1994 IBM Corporation

CICS Family Client/Server Support

Data Translation

Transparency

Bi-Directional

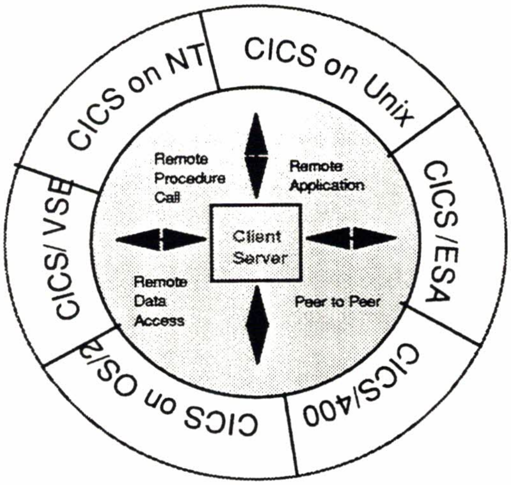
Security

System and Application Level Interfaces

Data Integrity

Route Through Intermediate Servers

Servers can be Clients



CICS on NT
CICS on Unix
CICS/ESA
CICS/400
CICS on OS/2
CICS/VSE
Remote Procedure Call
Remote Application
Client Server
Remote Data Access
Peer to Peer

"Open-Ended Client/Server"

CICS Application Server

Copyright 1994 IBM Corporation

CICNF410

CICS Client/Server Security

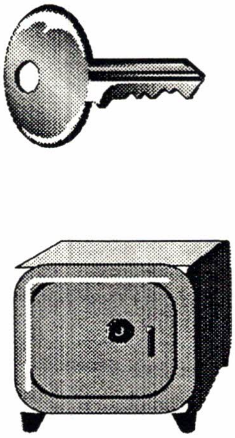
Sign-on
- CESN Program
- Program Interface

Resource Protection

Client/Server
- Communications
- Link
- Resource/Request

CICS Clients

Requests include User, Password



CICS Application Server

Copyright 1994 IBM Corporation

CICNF520

Application Development Tools

Advanced AD Tools

Dialog System
Visual Basic
VisualAge
Lotus Notes
PACBASE/CS
Flowmark

KASE: VIP
Power builder
PARTS
VisualGen
ADW
Early, Cloud's MDp

CICS Client

CICS
Application Server

COBOL
C
PL/I
REXX
C++

CICS Server

Established Third-Generation Languages

AD Tools

Customer References

Account	Industry	Country	Video	Published
CICS/6000				
Lamonts	Retail	US	Yes	Brief, Transaction
MEMC	Manufacturing	US		Brief, Transaction
JCIC	Finance	Taiwan		Brief
Dunn & Bradstreet	Business Services	US		Will be a brief
Barclays Bank	Finance	UK		Will be a brief
Asea Brown Boveri	Construction	Sweden		Will be a brief
Consolidated Freight	Distribution	US		Will be a brief
TVSM	Publishing	US		Will be a brief
CICS OS/2				
Olympic Games	Sports	Norway	Yes	
Petro Canada	Distribution	Canada		Brief
Banco Continental	Finance	Peru		Will be a brief, THINK
NTAA	Government	Japan		Transaction
Credit Mutuel de Bretagne	Finance	France	Yes	Client-Server News
Asea Brown Boveri	Construction	Italy		Will be a brief
US Trust Group	Finance	US		Will be a brief
FranenKische	Distribution	Germany		Will be a brief
Mass. Dept of Welfare	Government	US		Will be a brief
LANDP				
Thai Farmers Bank	Finance	Thailand		Will be a brief
CICS/400				
Black Hawk County	Local Government	US	Yes	Brief
SPRINT	Telecomms	US		Brief
Smith + Nephew	Manufacturing	UK		Brief,Transaction
VNIPI Sport	Sports Results	USSR		Transaction
SAUPIQUET	Manufacturing	France		Will be a brief
Laurier Life Insurance	Insurance	Canada		Will be a brief
CICS/ESA V4.1				
Standard Bank	Finance	S. Africa		Will be a brief
Southern Cal Edison	Utility	US		Will be a brief
MCI	Communications	US		Will be a brief
MQSeries				
Colonnial Williamsburg	Entertainment	US	Yes	Brief
Connecticut Mutal Life	Insurance	US	Yes	Brief, Transaction
Credit Suisse	Finance	Switzerland		Brief
Chicago Mercantile Exchange	Finance	US		Brief
CNRO	Pensions	France	Yes	
Barclays Bank	Finance	UK		Will be a brief
Pamuk Bank (DB2/6000)	Finance	Turkey		Will be a brief

1994 Winter Olympic Games - Lillehammer, Norway

PS/2's...RS/6000's...ES/9000's

- Results System
  - Capture and display results
  - Judges keyboards
- Info94 Application
  - Electronic mailbox
  - Games Management
- Based on CICS OS/2 and CICS/ESA

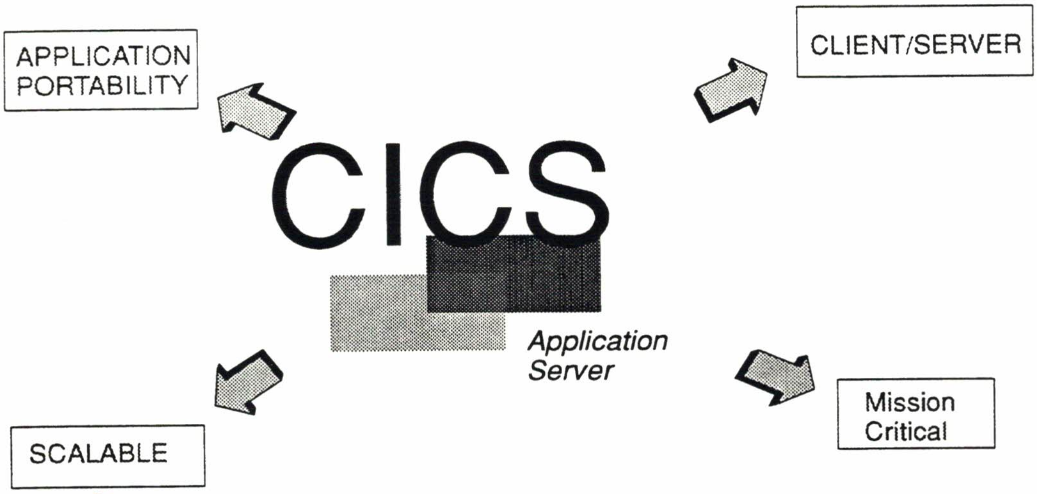
Gold, Silver, Bronze... and Blue...
and CICS too!

CICS Application Server

Copyright 1994 IBM Corporation

CICNF700

CICS Family - Scalable Mission Critical Client/Server



APPLICATION PORTABILITY

CLIENT/SERVER

CICS Application Server

SCALABLE

Mission Critical

- ROBUST, PROVEN APPLICATION SERVER

CICS Application Server

Copyright 1994 IBM Corporation

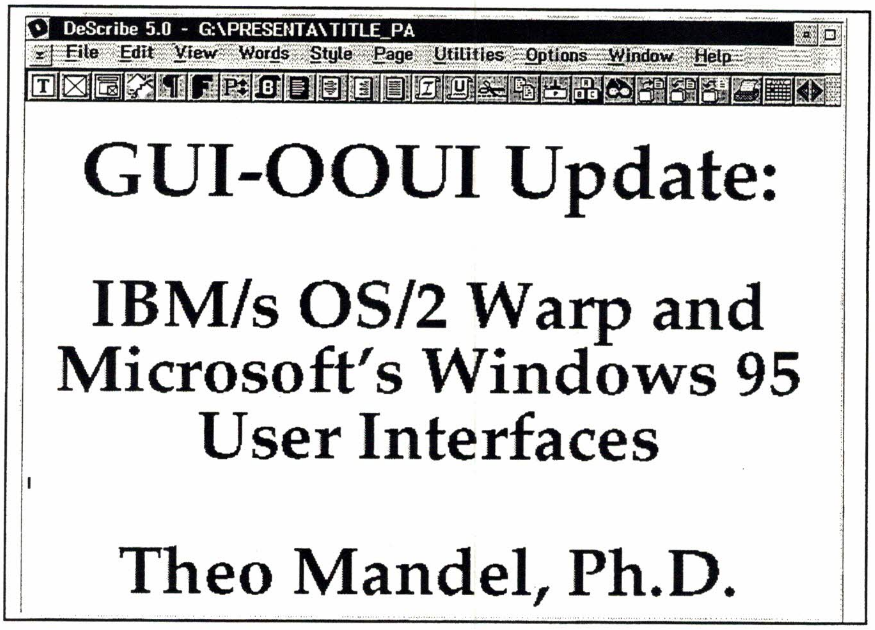
CICNF800

this page intentionally left blank

MGT 0604

GUI-OOUI Update: IBM's OS/2 Warp and Microsoft's Windows 95 User Interfaces

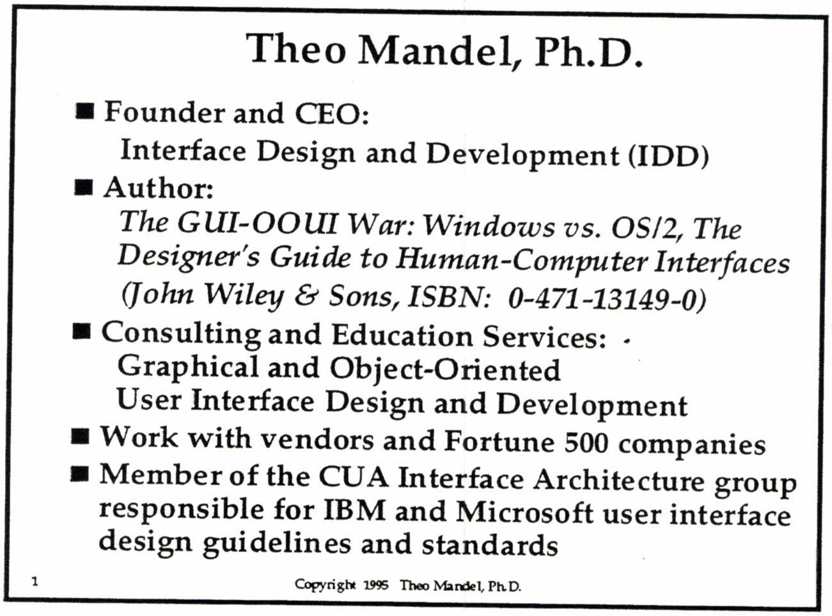
Theo Mandel
Interface Design and Development
Austin, TX


DeScribe 5.0 - G:\PRESENTA\TITLE_PA
File
Edit
View
Words
Style
Page
Utilities
Options
Window
Help

GUI-OOUI Update:

IBM/s OS/2 Warp and Microsoft's Windows 95 User Interfaces

Theo Mandel, Ph.D.



Theo Mandel, Ph.D.

- Founder and CEO:
  Interface Design and Development (IDD)
- Author:
  The GUI-OOUI War: Windows vs. OS/2, The Designer's Guide to Human-Computer Interfaces (John Wiley & Sons, ISBN: 0-471-13149-0)
- Consulting and Education Services:
  Graphical and Object-Oriented
  User Interface Design and Development
- Work with vendors and Fortune 500 companies
- Member of the CUA Interface Architecture group responsible for IBM and Microsoft user interface design guidelines and standards

1

Copyright 1995 Theo Mandel, Ph.D.

For Further Information, Please Contact:

Theo Mandel, Ph.D.
Interface Design and Development
8001 Two Coves Drive
Austin, Texas 78730-3125

(512) 345-2259 Phone/Fax

CompuServe: 75143,2160
Internet: 75143.2160@compuserve.com

Copyright 1995 Theo Mandel, Ph.D.

What to Expect from this Presentation

- What are the important user interface aspects of Windows 95 and OS/2 Warp
- Where user interfaces are going
- How these user interfaces affect you, your business, your developers, and your users

✓ ✓

Copyright 1995 Theo Mandel, Ph.D.

Why is the User Interface So Important?

"More than any other part of the operating system, the user interface defines a user's overall experience. The easier, more powerful, and more compelling the user interface, the better the user feels about computing, which, in turn, makes the user more productive."

"Great user interface helps grow the industry by making computing easier and more natural for all people, from the new user to the power user."

Microsoft Windows 95 Reviewer's Guide

Copyright 1995 Theo Mandel, Ph.D.

GUI

- Graphical User Interface
- Visual display of applications and data
- Icons represent applications
- Users must start applications first
- Provides functions to perform user tasks
- Related tasks are supported by other applications
- Training is centered on applications
- Users must follow application structure

OOUI

- Object-Oriented Interface
- Visual display of objects
- Icons represent objects
- Products are collections of cooperating objects & views
- Users open/work with objects, not applications
- Users more directly manipulate icons & objects
- Objects follow real-world metaphors and models
- Provides supplies and forms to perform user tasks
- Related tasks are supported by other objects

Copyright 1995 Theo Mandel, Ph.D.

Microsoft's GUI-OOUI Evolution

- Microsoft Windows User Interface Design Guide
  Preliminary Draft 4.00, May 1994

"The enhancements in the Windows interface pave the way for the evolution from basic graphical user interface design to a more object-oriented user interface design; that is, a more data-centric, rather than application-centric, interface...This is particularly important since this means from the user's perspective that applications become the engines behind the objects in the interface, enabling user interaction, but as transparent to users' tasks as the code that manages menus. This means that while applications will still be important, they no longer are the primary focus of the user. Users should be able to interact with their data without having to think about starting an application, allowing them to better concentrate on their tasks."

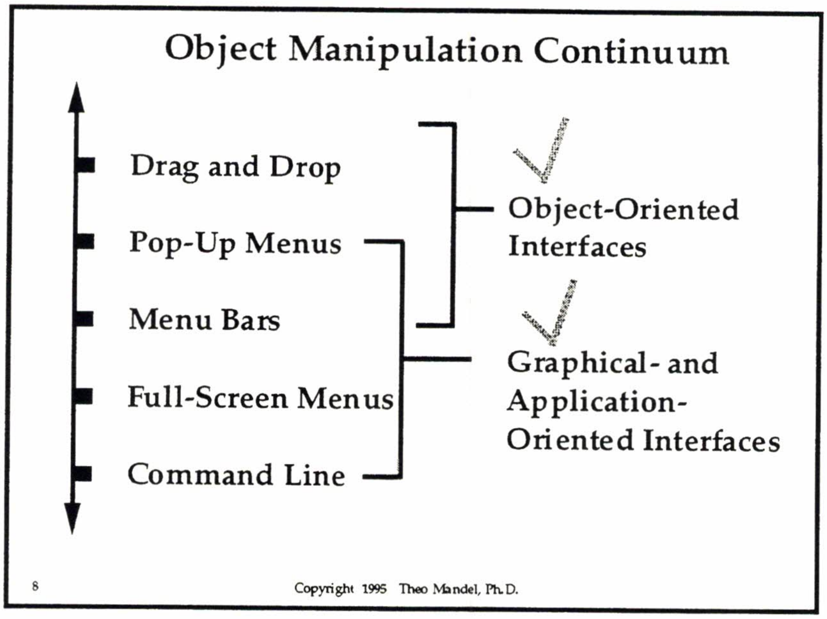
Copyright 1995 Theo Mandel, Ph.D.

User Interface Scorecards

✓✓

idd

Copyright 1995 Theo Mandel, Ph.D.



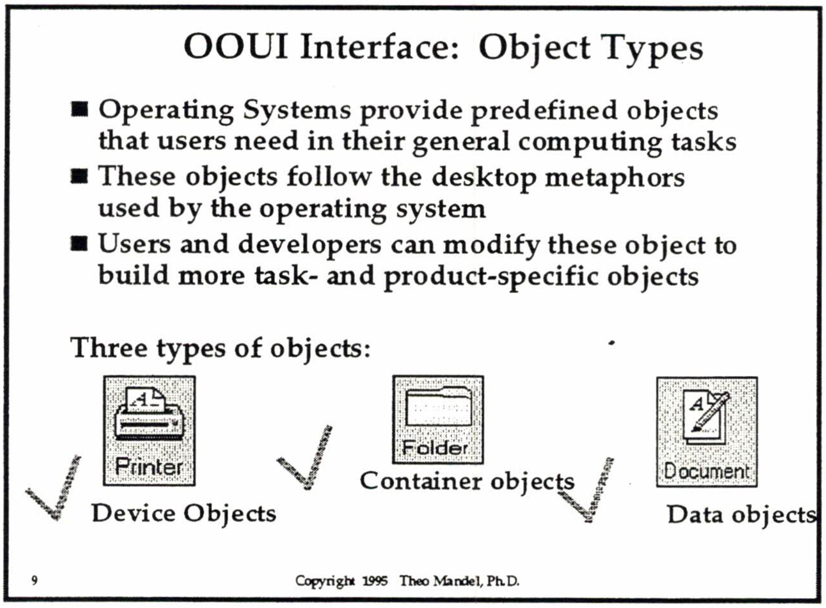
Object Manipulation Continuum

- Drag and Drop
- Pop-Up Menus
- Menu Bars
- Full-Screen Menus
- Command Line

✓ Object-Oriented Interfaces

✓ Graphical- and Application-Oriented Interfaces

Copyright 1995 Theo Mandel, Ph.D.



OOUI Interface: Object Types

- Operating Systems provide predefined objects that users need in their general computing tasks
- These objects follow the desktop metaphors used by the operating system
- Users and developers can modify these object to build more task- and product-specific objects

Three types of objects:

Printer
Folder
Document

✓ Device Objects

✓ Container objects

✓ Data objects

Copyright 1995 Theo Mandel, Ph.D.

OOUI Interface: Views of Objects

- ✓ FIRST: Take great care to determine WHAT objects users need
- ✓ SECOND: You must also determine HOW users need (and want) to work with the objects
  - What are the appropriate views?
  - What users need what views?
  - What views are used when?
  - What media should be used?
  - What business process and security measures affect objects and views?
  - Views should be user and context sensitive

Copyright 1995 Theo Mandel, Ph.D.

Golden Rules of User Interface Design

Major Categories:

✓ 1. Place Users in Control

✓ 2. Reduce Users' Memory Load

✓ 3. Make the Interface Consistent

See handouts for session OTCP07
"To GUI or to OOUI" for details

Copyright 1995 Theo Mandel, Ph.D.

User Interface Usability

✓ 1. **Installation**
   - -- Instructions, Prompts, Help
   - -- Customizable Install (Automatic, Advanced, Notebook,...)
   - -- User Interface Style (Text or Graphical)

✓ 2. **Ease of Learning**
   - -- Transfer of skills from DOS, Windows, OS/2
   - -- Tutorial, Help
   - -- Golden Rules of Interface Design

✓ 3. **Ease of Use**
   - -- Automation, Customization, Fast Paths
   - -- Golden Rules of Interface Design

Copyright 1995 Theo Mandel, Ph.D.

Golden Rules Scorecard

1. **Place Users in Control**

Low ←|——|——|——|——|→ High

2. **Reduce User's Memory Load**

Low ←|——|——|——|——|→ High

3. **Make the Interface Consistent**

Low ←|——|——|——|——|→ High

Copyright 1995 Theo Mandel, Ph.D.

Interface Usability Scorecard

1. Installation

2. Ease of Learning

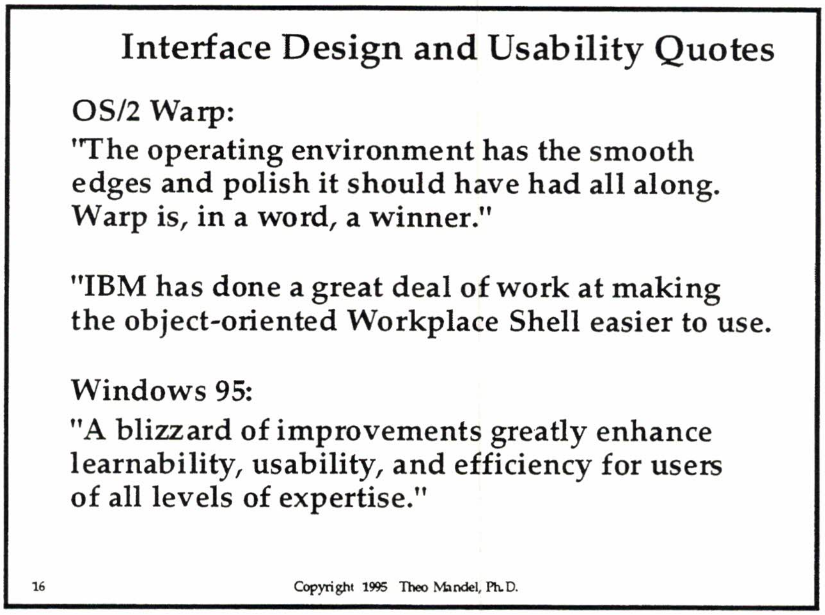
3. Ease of Use

Copyright 1995 Theo Mandel, Ph.D.

What Users are Promised
by
OS/2 Warp
and
Windows 95

idd

Copyright 1995 Theo Mandel, Ph.D.



Interface Design and Usability Quotes

OS/2 Warp:

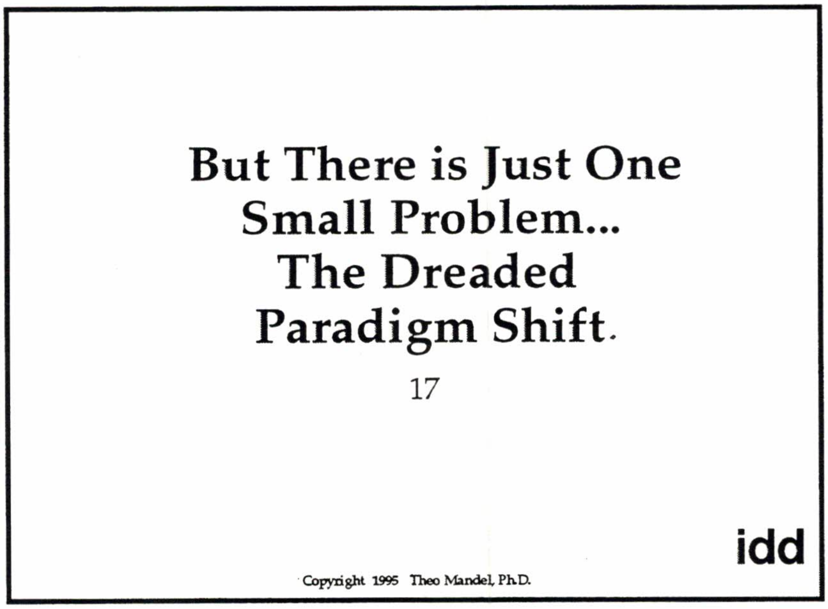
"The operating environment has the smooth edges and polish it should have had all along. Warp is, in a word, a winner."

"IBM has done a great deal of work at making the object-oriented Workplace Shell easier to use.

Windows 95:

"A blizzard of improvements greatly enhance learnability, usability, and efficiency for users of all levels of expertise."

Copyright 1995 Theo Mandel, Ph.D.



But There is Just One Small Problem... The Dreaded Paradigm Shift.

Copyright 1995 Theo Mandel, Ph.D.

The Paradigm Shift

OS/2 Warp:

"It still requires something of a paradigm shift to use, especially for experienced Windows users. But the more you use it, the more sense it makes."

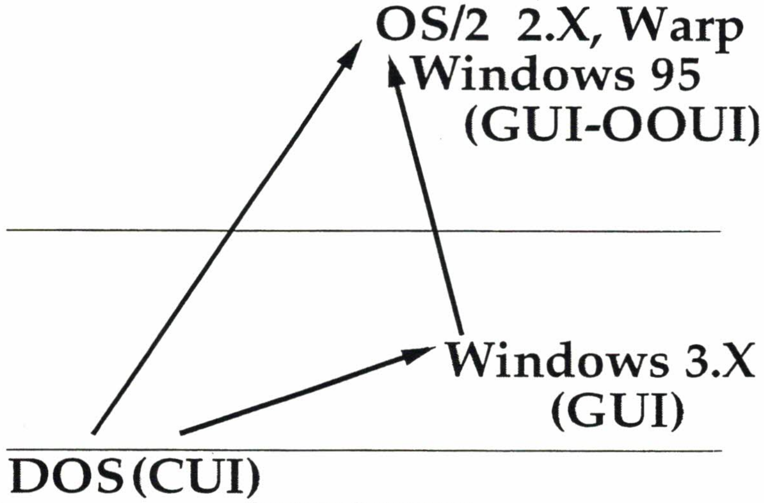
Windows 95:

"Efforts will be undertaken...to implement migration aids to make it easier for Windows 3.1 users to get up to speed with Windows 95."

"Once advanced and power users overcome the initial learning curve of the new Windows 95 user interface, they will become quickly familiar and comfortable with the additional functionality offered..."

Copyright 1995 Theo Mandel, Ph.D.

User Migration Paths



OS/2 2.X, Warp
Windows 95
(GUI-OOUI)

Windows 3.X
(GUI)

DOS (CUI)

Copyright 1995 Theo Mandel, Ph.D.

Solving the Paradigm Shift

The low-tech, easy-way-out solution to the paradigm shift problem:

"Here's how you can make the OS/2 Workplace Shell interface look and work just like Windows 3.1!"

Windows 95...watch out, the same thing will happen!

It's like taking a sports car and crippling it so it looks and works like a station wagon!

Copyright 1995 Theo Mandel, Ph. D.

OS/2 Warp
and
Windows 95
User Interfaces.

idd

Copyright 1995 Theo Mandel, Ph.D.

OS/2 Warp Interface Highlights

	Control	Memory	Consistency
■ LaunchPad	X	X	X
■ Interactive Tutorial	X	X	
■ Archive Desktop Config.	X	X	X
■ Animated Icons		X	X
■ New Color Schemes	X	X	X
■ New Color Palettes	X	X	X
■ New Object/File Finder		X	X
■ Folder WorkAreas	X	X	X
■ Object Templates	X	X	X
■ Close Parent Folders	X	X	
■ Overlap Window context	X		
■ Shadows of Objects	X	X	X

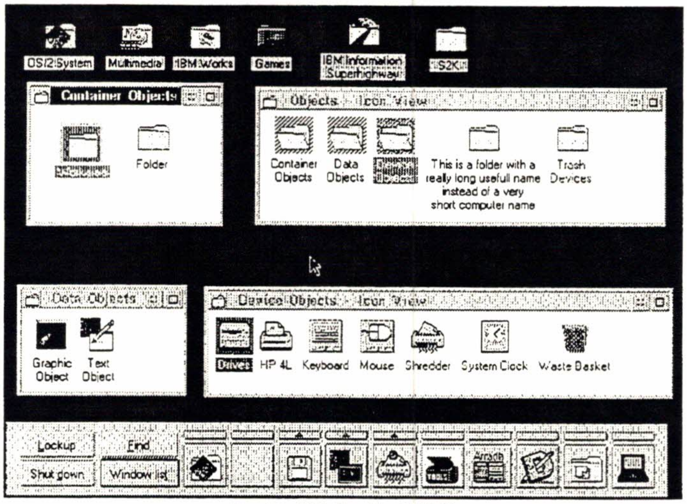
Copyright 1995 Theo Mandel, Ph. D.

OS/2 Warp Interface Highlights

	Control	Memory	Consistency
■ Object Pop-up Menus	X	X	X
■ Desktop Shredder object	X		X
■ Sticky Drag and Drop	X	X	X
■ Always Visible Icon Text		X	
■ Long File Names	X	X	X
■ Settings View of Objects		X	X
■ Application integration with OS/2 Desktop	X	X	X
■ Includes BonusPak of Integrated Applications	X	X	X
■ Settings Choice on Pop-up Menu	X X!!	X	
■ Undo Arrange Desktop	X!!	X	

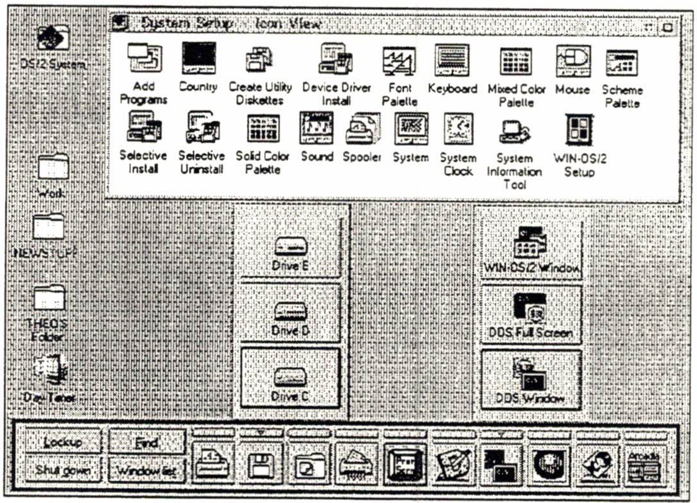
Copyright 1995 Theo Mandel, Ph.D.

OOUI Desktop: OS/2 Warp


OS/2 System
Multimedia
IBM Works
Games
IBM Information Superhighway
S2K
Container Objects
Folder
Objects - Icon View
Container Objects
Data Objects
This is a folder with a really long useful name instead of a very short computer name
Trash Devices
Data Objects
Graphic Object
Text Object
Device Objects - Icon View
Drives
HP 4L
Keyboard
Mouse
Shredder
System Clock
Waste Basket
Lockup
Find
Shut down
Window list

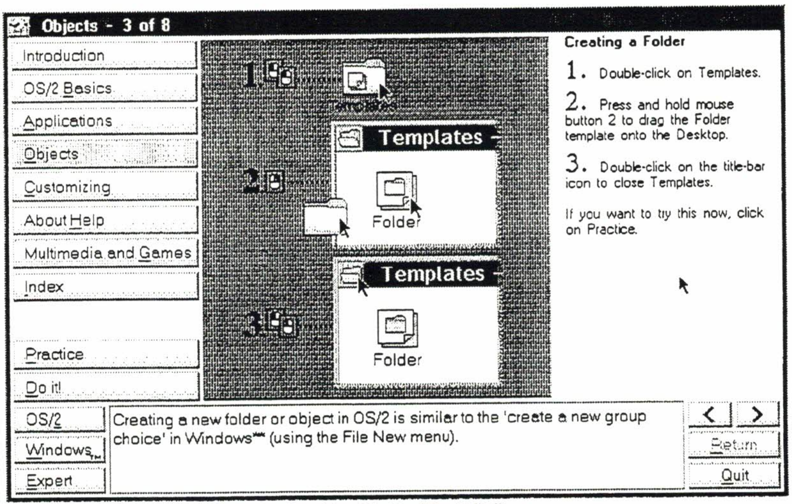
Copyright 1995 Theo Mandel, Ph.D.

OS/2 Warp User Inteface


OS/2 System
System Setup - Icon View
Add Programs
Country
Create Utility Diskettes
Device Driver Install
Font Palette
Keyboard
Mixed Color Palette
Mouse
Scheme Palette
Selective Install
Selective Uninstall
Solid Color Palette
Sound
Spooler
System
System Clock
System Information Tool
WIN-OS/2 Setup
Work
NEWSTUFF
THEOS Folder
Day Timer
Drive E
Drive D
Drive C
WIN-OS/2 Window
DOS Full Screen
DOS Window
Lockup
Find
Shut down
Window list

Copyright 1995 Theo Mandel, Ph.D.

OS/2 Warp User Inteface


Objects - 3 of 8
Introduction
OS/2 Basics
Applications
Objects
Customizing
About Help
Multimedia and Games
Index
Practice
Do it!
OS/2
Windows™
Expert
1.
2.
3.
Templates
Folder
Templates
Folder
<
>
Return
Quit

Creating a Folder

1. Double-click on Templates.
2. Press and hold mouse button 2 to drag the Folder template onto the Desktop.
3. Double-click on the title-bar icon to close Templates.

If you want to try this now, click on Practice.

Creating a new folder or object in OS/2 is similar to the 'create a new group choice' in Windows™ (using the File New menu).

Copyright 1995 Theo Mandel, Ph.D.

OS/2 Warp Summary

- OS/2 Warp gives you enhancements to an existing Object-Oriented user interface. It is a sophisticated, well-tuned user interface for all levels of users.
- OS/2 Warp is a fully object-oriented interface where applications can integrate with the Workplace Shell desktop.
- The Tutorial is an excellent migration and training aid for DOS and Windows users.
- The BonusPak contains OS/2 integrated object-oriented programs, helping users learn to use the interface.
- OS/2 Warp hides the underlying operating system well, but at the same time keeps the interface and the operating system in synch with each other.

Copyright 1995 Theo Mandel, Ph.D.

Windows 95 Interface Highlights

	Control	Memory	Consistency
■ Taskbar: Start Button	X	X	X
■ Taskbar: Task switching	X	X	X
■ My Computer	X	X	X
■ Network Neighborhood	X	X	X
■ New Help Interface	X	X	
■ Help Shortcut Buttons	X	X	
■ Create New Objects	X	X	X
■ Wizards		X	X
■ Explorer	X	X	
■ Object Pop-up Menus	X	X	X
■ Briefcase	X	X	X
■ Multilevel Undo	X	X	X

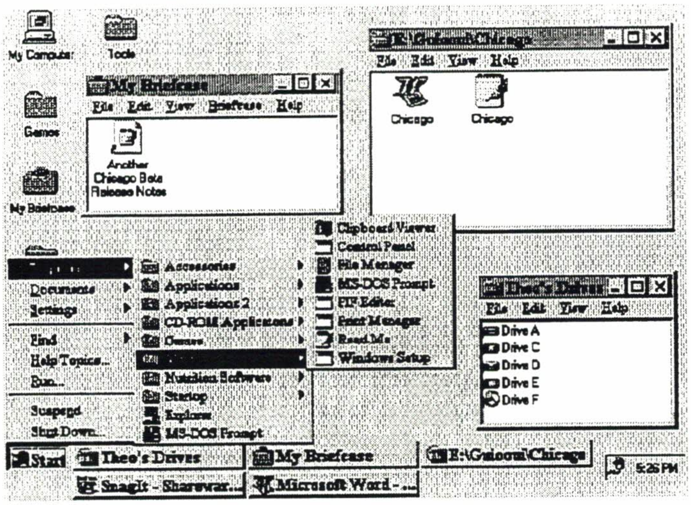
Copyright 1995 Theo Mandel, Ph.D.

Windows 95 Interface Highlights

	Control	Memory	Consistency
■ Recycle Bin	X	X	X
■ New Title Bar Buttons		X??	X??
■ Default and Non-	X	X	X
Default Drag	X	X	X
■ New Find Utility		X	X
■ Long File Names	X	X	X
■ Window Views Toolbar	X	X	X
■ Object Property Sheets		X	X
■ Object Shortcuts	X	X	X
■ Quick View Files		X	X
■ Customizable access for handicapped use	X	X	X

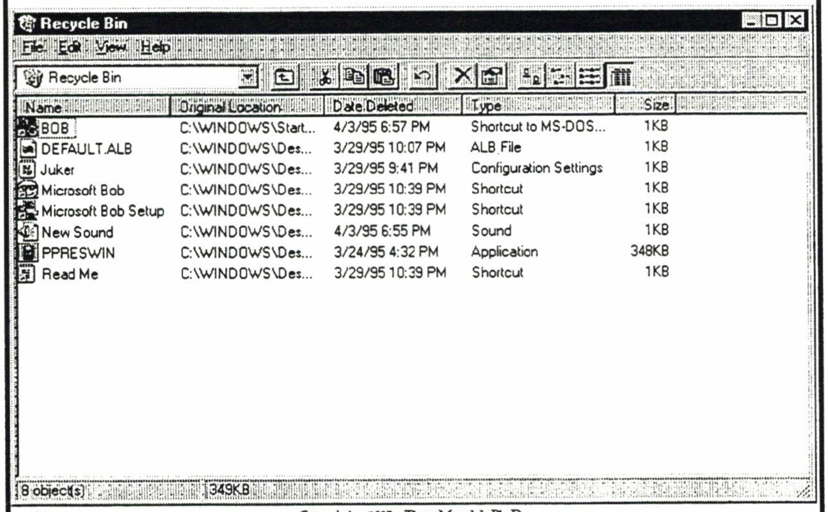
Copyright 1995 Theo Mandel, Ph.D.

Windows 95 User Interface


My Computer
Tools
Games
My Briefcase
My Briefcase
File Edit View Briefcase Help
Another Chicago Beta Release Notes
E:\Galoom\Chicago
File Edit View Help
Chicago
Chicago
Documents
Settings
Find
Help Topics...
Run...
Suspend
Shut Down...
Accessories
Applications
Applications 2
CD-ROM Applications
Games
Nutrition Software
Startup
Explorer
MS-DOS Prompt
Clipboard Viewer
Control Panel
File Manager
MS-DOS Prompt
PIF Editor
Print Manager
Read Me
Windows Setup
Theo's Drives
File Edit View Help
Drive A
Drive C
Drive D
Drive E
Drive F
Start
Theo's Drives
My Briefcase
E:\Galoom\Chicago
SnagIt - Shareware...
Microsoft Word - ...
5:26 PM

Copyright 1995 Theo Mandel, Ph.D.

Windows 95 User Interface


Recycle Bin
File Edit View Help
Recycle Bin

Name	Original Location	Date Deleted	Type	Size
BOB	C:\WINDOWS\Start...	4/3/95 6:57 PM	Shortcut to MS-DOS...	1KB
DEFAULT.ALB	C:\WINDOWS\Des...	3/29/95 10:07 PM	ALB File	1KB
Juker	C:\WINDOWS\Des...	3/29/95 9:41 PM	Configuration Settings	1KB
Microsoft Bob	C:\WINDOWS\Des...	3/29/95 10:39 PM	Shortcut	1KB
Microsoft Bob Setup	C:\WINDOWS\Des...	3/29/95 10:39 PM	Shortcut	1KB
New Sound	C:\WINDOWS\Des...	4/3/95 6:55 PM	Sound	1KB
PPRESWIN	C:\WINDOWS\Des...	3/24/95 4:32 PM	Application	348KB
Read Me	C:\WINDOWS\Des...	3/29/95 10:39 PM	Shortcut	1KB

8 object(s) | 349KB

Copyright 1995 Theo Mandel, Ph.D.

Windows 95 Summary

- Windows 95 is Microsoft's first attempt at a more object-oriented user interface.
- Windows 95 is a tremendous improvement over the Windows 3.1 user interface.
- Microsoft's initial efforts are focused at making Windows 95 easy-to-use for new and novice users.
- Experienced Windows 3.1 users will first experience the "Paradigm Shift". After continued use, however, users will find a more object-oriented user interface to be something they don't want to give up.
- It is too early to see how well Software Vendors follow the object-oriented interface style and how well they can build actual objects that reside on the user's desktop.

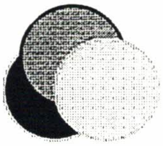
Copyright 1995 Theo Mandel, Ph.D.

Future User Interfaces: Beyond the Desktop

Copyright 1995 Theo Mandel, Ph.D.

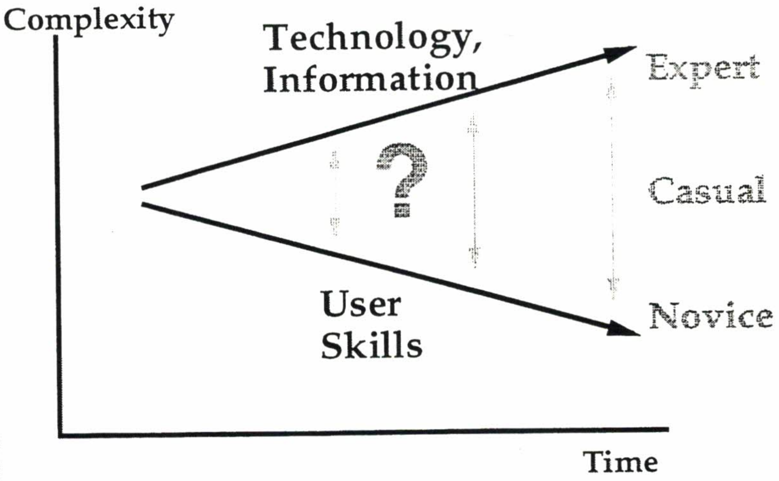
Future User Interfaces

- "The best way to predict the future is to invent it."
- Worst way to predict: Extrapolate from today
- Will future interfaces follow a desktop metaphor?
- Future user interfaces will evolve from:
  - Computing
  - Publishing
  - Entertainment



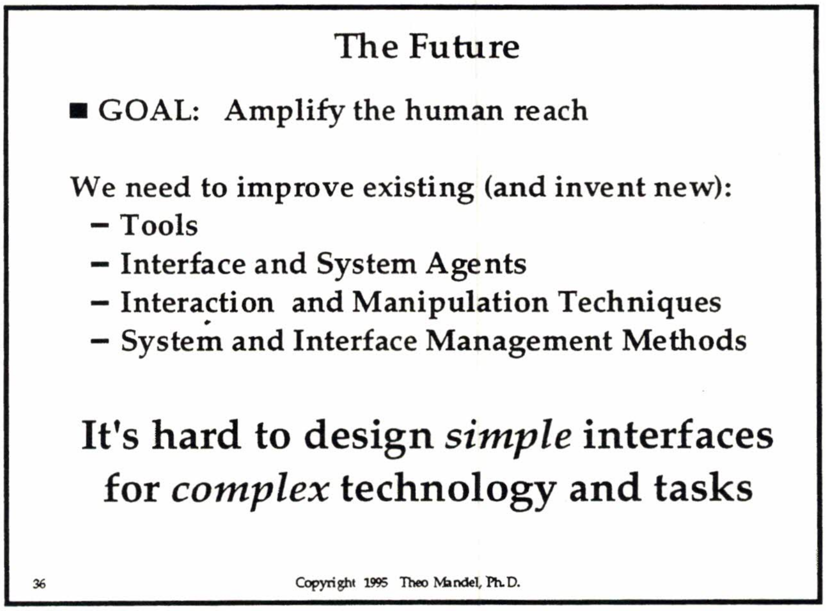
Copyright 1995 Theo Mandel, Ph.D.

The Technology Gap



Complexity
Technology,
Information
?
User
Skills
Expert
Casual
Novice
Time

Copyright 1995 Theo Mandel, Ph.D.



The Future

- GOAL: Amplify the human reach

We need to improve existing (and invent new):
- Tools
- Interface and System Agents
- Interaction and Manipulation Techniques
- System and Interface Management Methods

It's hard to design *simple* interfaces for *complex* technology and tasks

Copyright 1995 Theo Mandel, Ph.D.

MGT 0608

IBM Object Strategy and Plans

Anthony Brown
IBM

IBM OBJECT STRATEGY

The information processing industry continues to be in crisis. The reality of today's environment is that people more and more are using their information systems as a means of gaining a competitive edge, providing the best productivity, the fastest performance, or the cheapest cost. Unfortunately, the result is an explosion of disparate systems, each one an attempt to help each individual department meet their specific goals, resulting in a growing number of compatibility and interoperability problems. These problems are compounded significantly as we look at the applications we're running on our systems. Although companies are quickly trying to be innovative with their solutions, we cannot escape the fact that we are living in a legacy world.

According to the Gartner Group, there are 1 billion lines of COBOL code worldwide and over 1 trillion dollars invested in COBOL code executions in the US alone. Approximately 80% of all commercial applications are written in COBOL. In fact, a recent Gartner Group report stated, that even if you wanted to start over it would take developers around the world, working full time, 62.5 years to replace all of the legacy applications that exist today. As a result of these pressures, traditional development methodologies are no longer adequate to address the changing needs of today's environment. IBM is committed to providing the most comprehensive set of products and services to our customers. A key element of these solution services is IBM's commitment to object technology.

A TIME OF TRANSITION

We are witnessing the struggle to make a transition. A transition to move from a labor-based environment, to one that is asset-based. This transition can be stated several ways. Transition from individual skills to organizational competence, from initial OO exploitation to company-wide object deployment, from labor-intensive environments to an enterprise asset base. The labor-based approach involves many people continually rewriting the same code. The approach can be best characterized by a lack of standardization, no leveraging of assets and an enormous drain on productivity. By contrast, the asset-based approach involves re-using the same pieces of code. The asset-based approach is characterized by standardization, integration, leverages assets and increased productivity.

COMPLETING THE EVOLUTION

This transition is the final step in the evolution we've all seen in this industry. The evolution began with the first computers. Every application was dependent upon the hardware it ran upon. Hardware was the determining factor. Eventually the industry moved away from a hardware dependency to one that was highly dependent on the operating system and later dependent on the specific application that was being run. Now the focus is switching away from the technology and concentrating on what we want the technology to do for us, how to work better, smarter, and faster.

The best way to make technology work for us is to make it as simple as possible to use. Strength lies in assets that simplify the complexity of our computing environment.

OBJECT TECHNOLOGY DRIVERS

The drivers to make this transition are coming from three primary sources, the end-user, the developer community and the organization itself. The end user wants better access to information. The notion of information anytime, anywhere. In addition, a shift away from traditional application-based environments to activity-based computing. Essentially, end-users are looking for faster, more innovative solutions to their problems. In support of end-user needs, application developers are looking for improved developer productivity, open and extensible architectures, and industry standards compliance. The drive toward reusable components, which will enable all of the above. The global organizational perspective is a combination of the earlier two, an increased drive for an overall reduction in development costs, distributed information, and heterogeneous environments that seamlessly work together. Solutions adaptive to their specific needs.

OBJECT-ORIENTED DEVELOPMENT

In response to these drivers, companies are beginning to realize the true value of OO technology. Adoption rate is accelerating. Recently, the Meta Group reported, 30% of new application development will be object-based by 1999, 70% in 2003. In addition, 50% of COBOL users are now experimenting with or already using some form of object technology. Exciting as this technology is, organizations have a need for ways to leverage this emerging technology while maximizing their current investments

THE VALUE PROPOSITION

IBM has developed its object technology strategy based on a value proposition to our customers which provide the necessary tools that improve developer productivity, help customers deliver superior applications that enhance their competitive edge and provide the ability for them to deliver those applications on a variety of systems. Overall, this value proposition focuses on leveraging their assets, skills of end-users and developers, current applications and current systems infrastructure. This proposition must accomplish all of this in a way that provides solutions to key application development issues, faster development time, portability to the platforms they use today and may want to use tomorrow, and provide easy maintenance of new and existing applications.

A COMPLETE SET OF APPLICATION DEVELOPMENT OFFERINGS

This proposition begins with a complete set of applications development offerings, designed to help our customers begin with their existing environment and at their own pace, migrate toward an object-oriented environment. IBM's application development offerings encompass all phases of the application development process, design and model applications, build in the language of your choice and the ability to manage, maintain and control your AD environment.

THE ASSET BASE

With these tools customers can establish an asset base to build and grow from. The asset base is built from objects, class libraries, and frameworks. Class libraries provide you an extensive choice of building blocks for creating your applications, handle the complexity of low-level APIs so you can focus on your application and provide a foundation for visual builders. Frameworks on the other hand provide a higher level of abstraction. Frameworks are related classes designed to work together, providing generalized solutions to complex problem domains. IBM is making many contributions in the these areas, in particular, IBM OpenClass for class libraries, VisualAge and OpenDoc for components, Taligent and OpenDoc for frameworks and Object Connection for vendor programs.

GROWING THE ASSET BASE

Growing the asset base is key to protecting our customers' investments in information systems. The ability to use and extend the capability of assets is also key to fulfilling our value proposition to our customers. The strategy for growing the asset base requires developing a library of assets that are easy to build or buy, easy to modify, and easy to assemble and use. These assets should be implemented in a way that leverages existing assets. These

assets come from a variety of sources;
system vendors, software vendors, frameworks like CommonPoint and IBM invention. Assets are made up of class libraries and components which capture business logic and application dependencies unique to a particular installation. Initial implementations focus on a language-based way to deliver C++ and Smalltalk classes. Over time these implementations must evolve to provide a variety of functions, including: GUIs, data access, transaction processing, communications protocols, business logic, and advanced functions like multimedia. In addition, these implementations must also provide the capability to be managed through a central repository. Growing the asset base is a means to reduce the skills requirement, dramatically improving the build process and dramatically improving productivity.

THE RIGHT SET OF TOOLS FOR THE JOB

The bottomline is that customers need the right set of tools for the job of developing applications. One of the biggest benefits IBM has to offer is its ability to provide interoperability in the truest sense of the word. Interoperability which lets customers develop and maintain applications faster, lets them continue to work with existing platforms and lets them leverage developer skills and knowledge base. That means customers can continue to develop in a traditional environment, begin to use visual construction from parts, move into a more team-driven development approach, and finally, bring it all together to support the development of distributed applications.

IBM OBJECT TECHNOLOGY ROADMAP

In the next year, IBM is committed to deliver the best tools, services, and education in the industry. Today, we already offer robust client/server object systems and languages, tools like the OpenDoc and SOMobjects Developers Toolkit, language support like C++, Smalltalk and OO COBOL. In addition, we are providing extensive class libraries support through OpenClass and Smalltalk classes. By the end of this year, you'll see us complete the transition to open, reusable component environments, including distributed Smalltalk, OT Analysis/DES tool, enhanced OpenClass, OpenDoc on OS/2, Win and Mac, workgroup parts for OpenDoc, visual environment for Basic and AIX CommonPoint.

And by 1996, our customers will be able to reach the full potential of an enterprise-wide, object-oriented approach. Including integrated modeling, extended platform support, additional OMG services, Cobra 2.0 compliance and Parts availability. And of course, all of our tools and technologies will be enhanced by a broad spectrum of consulting, services, and education.

POINTS OF ENTRY

SOM

Underlying the IBM object strategy is an evolutionary process. We began by defining the structure necessary to make object technology viable in today's world of diverse applications, based on the IBM System Object Model (SOM). SOM is an advanced object enabling technology designed specifically to address the need for an open, flexible software infrastructure to facilitate software component integration. SOM defines an infrastructure for sharing objects. It allows objects to be packaged in a way that exposes only their interfaces. As a result, objects and classes can be built which are language and operating system neutral. In addition, SOM scales to support objects that are in a distributed system through the widely accepted distributed SOM services. Distributed SOM is a standards-based deployment for distributed services based on the Object Management Groups CORBA standard. Distributed SOM services locate the remote object and route requests and responses, masking this complexity from the developer and the end-user. IBM plans to make distributed SOM available on all major platforms.

OPENDOC

Components are now being generated under a powerful industry initiative called OpenDoc. OpenDoc allows users to view and edit text, graphics and multi-media content using individual application components, without the overhead of invoking an entire separate application. OpenDoc consists of three well-established industry technologies. OSA, an Open Scripting Architecture that allows vendors to plug in any of their scripting products to automate the use of object technology. BENTO Storage technology which allows you to store not just the content of objects but the also the descriptions of them. Bento also provides the ability for components to be shared dynamically. The System Object Model (SOM), provides the default dynamic object linking mechanism for Opendoc.

OpenDoc makes off-the-shelf component parts a viable proposition. The OpenDoc consortium is managed by Component Integration Laboratories which provides reference source code, technical documentation, example software and validation services. OpenDoc also provides an entry level path to CommonPoint.

COMMONPOINT

CommonPoint provides a set of rich-function object-oriented frameworks which broaden IBM's already extensive portfolio of application development products. The CommonPoint frameworks and their associated development environment are designed to enable a new generation of collaborative applications to be built, customized and deployed on powerful desktop systems and workstations. CommonPoint frameworks allow the creation of application solutions that share data, information and services across distributed networks. With CommonPoint, developers can focus on strategic, business-specific aspects of their development efforts, rather than the technical complexities and maintenance of the design environment. The objective of CommonPoint, is to produce a robust enterprise-wide computing environment that protects the customer investment in current infrastructure and applications, while providing the benefits of a fully object-oriented environment.

IBM is committed to providing the most comprehensive set of leading-edge object technology products and services in the industry. These vary in complexity and capability, allowing developers to select the most suitable point of entry into object programming based on their skill sets, their existing tools and technologies, and the needs of their customers.

PRO 0602

An Overview of Object COBOL for OS/2

Frank Coyle

Dallas, Texas

Dr. Frank P. Coyle
Computer Science Department
Southern Methodist University
Dallas, TX 75240

As we move forward into the 21st century, there are two hard realities. The first is the inevitable move to object technology. Object technology is a natural match for the details of client-server computing and provides a consistent thread, that of objects, throughout the software lifecycle. The second reality is that the billions of lines of COBOL code, which support the infrastructure of countless corporations, will not go away. Far too many organizations have too much invested in reliable, tested and well-functioning COBOL code to risk a major upheaval in their business. For many organizations, Object-Oriented COBOL will be the obvious choice for dealing with this reality and OS/2 will be positioned to facilitate the change.

In May 1995, the ANSI Object-Oriented COBOL Task Group (OOCTG) released a document that establishes a baseline for the Object-Oriented COBOL language until 1997 when ANSI will officially lock in the standard. However, organizations with major investments in COBOL and a work force of experienced COBOL programmers are now able to gain competitive advantage by the gradual transition to Object-Oriented COBOL without abandoning or totally reworking their legacy systems. Object-Oriented COBOL is intended to be 100% compatible with the existing ANSI-COBOL standard and allows the gradual introduction of object-oriented features and stepwise modification of existing legacy systems. Thus, organizations will not be forced to rewrite all their code as a consequence of adopting object technology. Object-Oriented COBOL will provide a path for the gradual transition to object-oriented thinking and in-house training. The three major players in this burgeoning market - IBM, Hitachi and MicroFocus are now delivering Object-Oriented COBOL compilers to address these requirements and to help organizations meet the challenges of 21st century computing.

THE OBJECT-ORIENTED COBOL LANGUAGE.

Object-Oriented COBOL directly supports object-oriented concepts by using class, object, and inheritance in a manner logically consistent with the established terminology. Semantically, only the focus of abstraction changes from analysis through implementation. Object-Oriented COBOL is consistent at the implementation phase of development with concepts introduced during the object-oriented analysis phase. Classes, objects, inheritance, methods, attributes, and relationships, all fundamental to object-oriented analysis and design, are conceptually and computationally equivalent in Object-Oriented COBOL with other object-oriented languages.

Object-Oriented COBOL compliments the use of object-oriented analysis and object-oriented design, and supports object-oriented programming through the addition of the following capabilities to the COBOL language:

* The ability to define classes, comprising class object definitions and object definitions.
* The ability to define data encapsulated inside class objects and objects.
* The ability to define methods for class objects and objects.
* The ability to use inheritance and define subclasses.
* The ability to use polymorphism and interfaces for maximum flexibility.
* The ability to define data items able to hold references to objects.
* The ability to invoke methods on objects.

However, the new object-oriented features do not change the basic structure of a COBOL program. Object Oriented COBOL remains a combination of Identification, Environment, Procedure, and Data Divisions. However, several new sections, paragraphs, and reserved words have been added to the language. The following sections provide an overview of some of these major features..

CLASS DEFINITION.

The class is the basic component of OOCOBOL and the Class-Id paragraph begins a class definition. The class definition provides the structure for describing class variables, instance variables, class and object methods, and the position of the class in the inheritance hierarchy. The Class-Id paragraph lets both the reader and the compiler know that the complete definition of a class is about to begin. Each class in an Object-Oriented COBOL program has its own Class-Id paragraph. However, there are two aspects to a class - the class as a factory for instance objects and the class as definer for what the individual instance objects will look like. Thus, an Object-Oriented COBOL class definition has two parts, (1) a Factory paragraph that specifies the data and methods a class factory will use to create instance objects and (2) an Object paragraph that specifies what data and methods the classes' object instances will have.

CLASS-ID.

A class definition in Object Oriented COBOL begins with an Identification Division that contains definitions between the Class-Id paragraph and the End Class delimiter. A Class Definition is a source unit (a syntactically correct set of COBOL statements) which optionally includes an Environment Division, a Factory Definition, and an Object Definition. The listing below includes the major constructs for defining a class, including its factory object, methods, and instance objects. A standard COBOL program is shown in comparison.

Object Oriented COBOL

```
[Identification Division.]
Class-Id. class-name-1 Is [attributes].
[Environment Division.]
[Identification Division.
Factory.
[Environment Division.]
[Data Division.]End Program.
[Procedure Division.
[{Methods}...]]
End Factory.]
[Identification Division.]
Object.
[Environment Division.]
[Data Division.]
[Procedure Division.
[{Methods}...]]
End Object.]
End Class.
```

COBOL

```
[Identification Division.]
Program-Id. prog-name Is [attributes].
[Environment Division.]
[Data Division.]
[[Procedure Division.]
{procedure division entries}]
```

CLASS FACTORY DEFINITION

The class Factory definition defines the methods (code) and data associated with object creation. Factory data is specified in the Factory's own Data Division and its methods are specified in the Factory's Procedure Division. By default, all classes in Object Oriented COBOL have the capability of creating object instances through the use of the Object-Oriented COBOL reserved word "new". Therefore, not every class definition requires a Factory method or Factory data - only classes that require special creation (constructor) or deletion (destructor) methods or that need to maintain data that is global to all object instances. If Factory object methods are not specified, default constructors and destructors are provided.

The Factory Definition of a class may include an Environment Division. A Factory's Environment Division applies to every method defined within the Factory Definition. The Factory Environment Division may also include an Input-Output Section. The Input-Output Section may contain a File-Control paragraph and a I-O-Control paragraph. However, the Environment Division of a Factory Definition is not permitted to contain a Configuration Section.

METHODS

To use an object it is necessary to communicate with the object's interface. Objects may have many interfaces because each object may have many methods. Messages sent to an object are received by a method's interface. Messages may have parameters which are specified in the Procedure Division header of each method. A Procedure Division header contains Using and Returning phrases (the parameter list) that specify the data-items a method can receive along with the data-name a method can return.

The following code fragment illustrates a method definition that contains a Procedure Division with Using and Returning phrases..

```
[Identification Division.]
Method-Id. method-name-1.
[Environment Division.]
[Data Division.]
Procedure Division Using... Returning.
  procedure division entries
Exit Method. [exit attributes]
End Method method-name-1.
```

Note that the Procedure Division header may include a Returning phrase. The Returning phrase allows at most one parameter of any valid data type to be returned when a method is invoked.

INVOKING METHODS

Methods are activated by sending a message to an object. In Object COBOL, the Invoke statement is used to send messages. The Invoke statement consists of the reserved word Invoke followed by a class-name, the method-name, and (optionally) the Using and Returning phrases. Identifiers optionally supplied with the Invoke statement must agree with the method's parameter list.

The following is an example of how a method is invoked.

```
INVOKE  Class-1 "Method-1" USING  Received-Parameter  RETURNING Returned-Parameter
END-INVOKE.
```

The following code fragments illustrates how to define a method in class-1 that invokes a method in class-2.

Invoking method

```
Class-Id. Class-1.
Identification Division.
Method-Id.  Method-1.
```

```
Data Division.
Working-Storage Section
01 A             Pic 9(02) value 5.
01 B             Pic 9(02) value 10.
01 C             Pic 9(02) value 0.
Procedure Division.
        Invoke Class-1 "Method-2"
                Using      A, B,.
                Returning        C.
        End-Invoke.
        If C = 15
        Then
                Continue
        Else
          Stop Run
 Exit Method.
```

Invoked method

```
Identification Division.
Method-Id. Method-2.
Data Division.
Linkage Section.
01 D                    Pic 9(02).
01 E                    Pic 9(02).
01 F                    Pic 9(02).
Procedure Division Using  D, E, Returning F.
Compute F = D + E.
Exit Method.
```

In the above example, a common message passing area is established by declaring the parameters of an Invoke operation in both the invoking and the invoked methods. For example, data-names A, B, and C are declared in the invoking method. These data-items are referenced by the data-names D, E, and F in the invoked method. The data-item referred to as A in Method-1 corresponds to data-item D of Method-2, B corresponds to E, and C corresponds to F. The data-names are user-defined and may or may not be the same in each method. Method-1 does not know or care what Method-2 names its data-items. The Invoke statement simply passes the contents, not the data-names to the invoked method. In this case, Method-1 passes the numeric values 5 and 10 to Method-2 and is returned the numeric value 15.

A Linkage Section must be declared in the invoked method to be able to receive and return parameters. The Linkage Section declares all of the parameters to be passed between the two methods. In addition, the Procedure Division header of the invoked method must contain the same data-names as declared in the Linkage Section.

The data-items passed from the invoking method must match the invoked method in:

1. number of data-items
2. data type
3. size

Invoke transfers the flow of control to a method in much the same way a Call transfers the flow of control to a subprogram. A Returning phrase may also be used in a Call.

SUMMARY

The above examples are intended to give the flavor of an Object-Oriented COBOL program. As we have seen, one does not have to learn a totally new language in order to make use of Object-Oriented COBOL. Significant effort has been made to make the new language compatible with the old and permit programmers, knowledgeable in COBOL and most importantly, the business application, to make the transition to object-oriented thinking. Because OS/2 is expected to play a major role in the client-server revolution, developers need to become aware of how Object-Oriented COBOL for OS/2 fits in the broader context of large-scale interoperability where mainframes running COBOL and its object-oriented descendants serve as data and object repositories. While vendors such as IBM, Hitachi and MicroFocus will evolve their Object-Oriented COBOL products over the next two years, the final outcome will be an ANSI standard in 1997 and, undoubtedly, a significant role for OS/2.

PRO 0609

Exploring Genetic Algorithms and Evoltionary Programming

Larry O'Brien
Miller Freeman, Inc.
San Francisco, Calif.

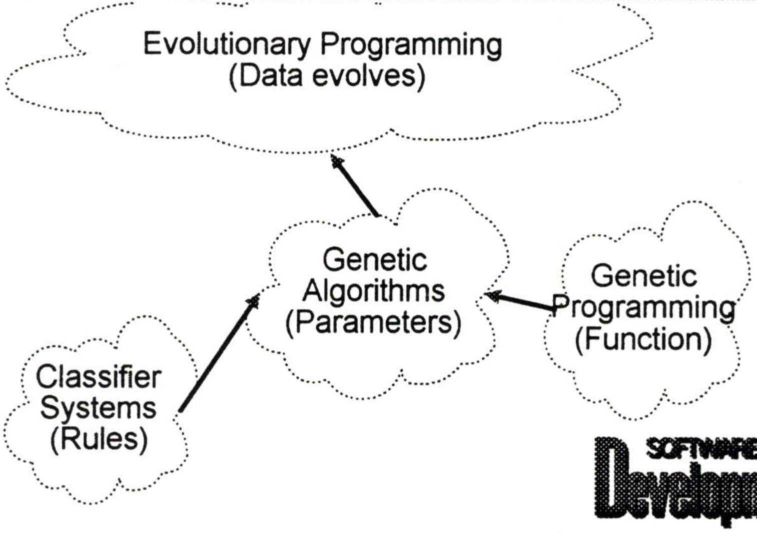
Implementing Genetic Algorithms

Larry O'Brien

Editorial Director
Software Development
Game Developer
AI Expert (ret.)
“Programming Artificial Life With MFC”

SOFTWARE Development

Evolutionary Programming Big Picture



Evolutionary Programming
(Data evolves)

Genetic
Algorithms
(Parameters)

Genetic
Programming
(Function)

Classifier
Systems
(Rules)

SOFTWARE Development

A Chicken Is An Egg's Way of Reproducing Itself

- Exactly!
- Life is the set of processes that sequences of nucleic acids use to create duplicate of themselves
- Competitive reproduction over time is the ultimate general-purpose optimizing algorithm
- Non-directed, parallel, robust

SOFTWARE Development

Evolution As Optimizer

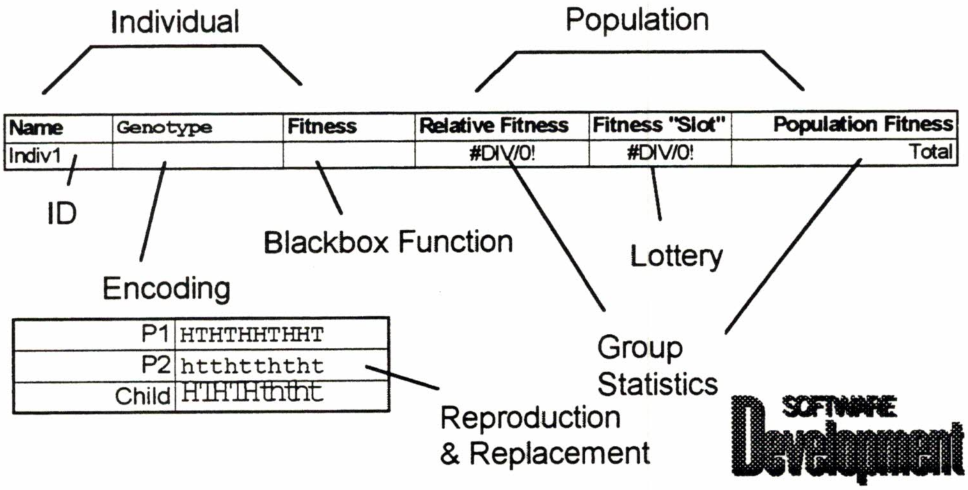
- Basic "blackbox" function of life is:
  - » Maintain my structure
  - » Eat
  - » Avoid being eaten
  - » Reproduce
- Competition for limited reproductive opportunities ("Survival of the Fittest")
- Mendelian Genetics

SOFTWARE Development

Let's Evolve

Demo

So What Did We See?

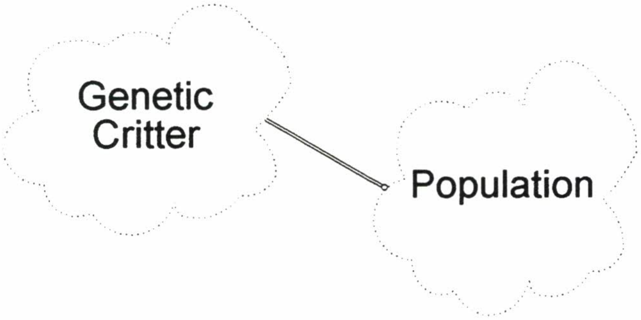


Name	Genotype	Fitness	Relative Fitness	Fitness "Slot"	Population Fitness
Indiv1			#DIV/0!	#DIV/0!	Total

P1	HTHTHHTHHT
P2	htthtththt
Child	HTHTHththt

Individual
Population
ID
Encoding
Blackbox Function
Lottery
Group Statistics
Reproduction & Replacement

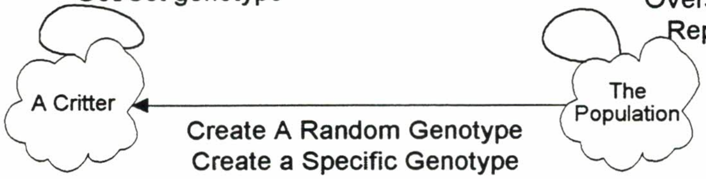
Two Obvious Classes



Genetic Critter

Population

What Goes On?



Translate genotype into phenotype
Evaluate phenotype as fitness
Get/Set genotype

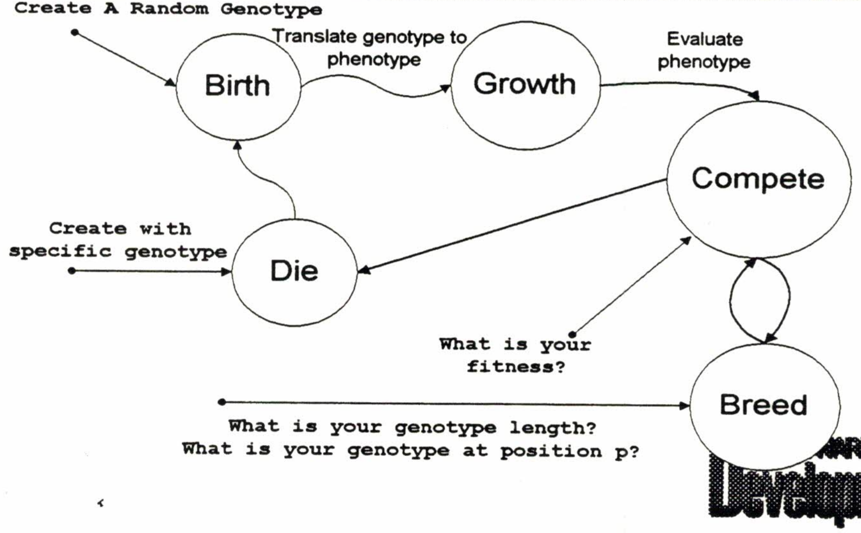
Initialize
Coordinate expression
Gather statistics
Perform lottery
Oversee breeding
Replace worst

A Critter

The Population

Create A Random Genotype
Create a Specific Genotype
What's Your Fitness?
Tell Me The Length of Your Genotype
What is your genotype at position p?

The Unromantic Life of a Genetic Critter



`Create A Random Genotype`

Birth

Translate genotype to phenotype

Growth

Evaluate phenotype

Compete

`Create with specific genotype`

Die

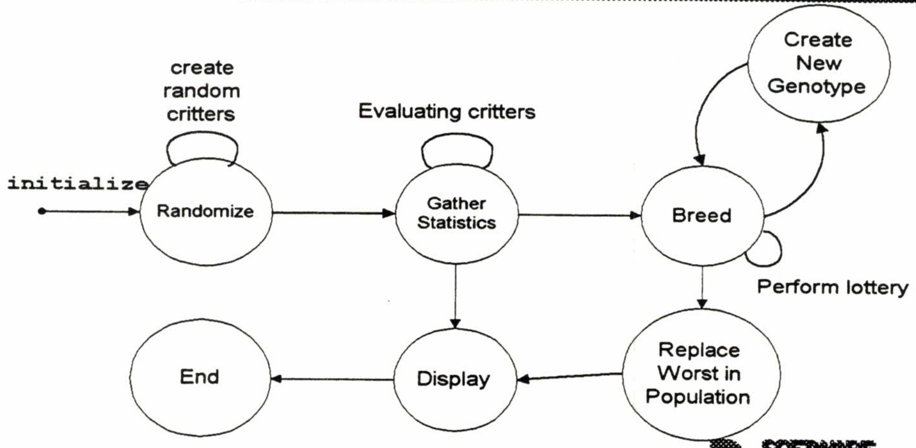
`What is your fitness?`

Breed

`What is your genotype length?`
`What is your genotype at position p?`

WARE Development

The Happy Life of the Population



`initialize`

create random critters

Randomize

Evaluating critters

Gather Statistics

Create New Genotype

Breed

Perform lottery

Replace Worst in Population

Display

End

SOFTWARE Development

Sidestepping Knowledge Representation

- "Classic" AI attempts to create an internal representation of the problem domain
  - » No epistemological nor empirical evidence for such a view
  - » (Also, they're hard)
- "Emergent" AI attempts to generate "good enough" solutions to the task at hand
  - » Not a sufficient answer to hard philosophical questions
  - » (But they work)

Mutation is Nothing

- Mutation is a fundamentally randomizing act
  - » Generally results in dimished fitness
  - » Introduces new low-fitness patterns for optimization
- Recombination is the engine of evolution
  - » Arranges small, "pretty good thoughts" into larger "great ideas"

Evolution as Schema Theory

- Genotype has small alphabet, fixed "word" size, long sentences.
    - » Nucleic acids: 4 letters taken 3 at a time encode 20 amino acids. AAs combine to make proteins.
        - – (Human genome information space: $4^{2,870,000,000}$)
    - » Typical GA uses binary alphabet (packed integers), encodes arbitrarily complex information.

Pause That Refreshes GA

After Goldberg, Koza (1990)

- Left bit: 0 = \$0.50, 1 = \$10.00
- Middle bit: 0 = Wine, 1 = Coca Cola
- Right bit: 0 = Leisurely, 1 = Fast Service
- Four restaurant's daily income:
    - » 011 → \$300 001 → \$100
      110 → \$600 010 → \$200

Searching for the Pattern

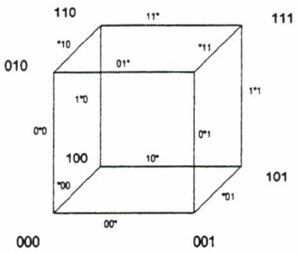
- Fitness of 110 is twice population average
  - » Is it the low price? 1**
  - » Is it the coca cola? *1*
  - » Is it the leisurely service? **0
  - » Is it the low price with coke? 11*
  - » etc
- Fitness of 010 is 2/3 population average
  - » Is it the high price? 0**
  - » Is it the coca cola? *1*
  - » Is it the leisurely service? **0
  - » Is it the high price with coke? 01*

SOFTWARE Development

Schema Specificity

Schema Specificity	Hyperplane Dimension	Geometric Realization	Individuals in schema	Number of such schema
3	0	Point	1	8
2	1	Line	2	12
1	2	Plane	4	6
0	3	Cube	8	1

Example Schema	Individuals
000	000
00*	000, 001
0**	000, 001, 010, 011
***	000, 001, 010, 011, 100, 101, 110, 111



SOFTWARE Development

Schema Theory (Long version)

(Goldberg, Koza 1990)

$$m(H,t+1) \geq m(H,t)\frac{f(H)}{\bar{f}}\left[1 - p_c\frac{\delta(H)}{l-1} - p_o(H)\right]$$

where

f – fitness

H – schema

m - number of schemata

δ – defining length

o – order

p_c - crossover probability

pm– mutation probability

l – string length

SOFTWARE
Development

Schema Theory (short version)

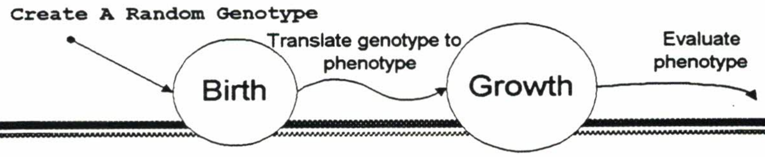
- Short (low specificity), relatively fit schema increase and combine and evolve into more specific, higher fitness schema.
- Trials are allocated proportional to fitness

SOFTWARE
Development

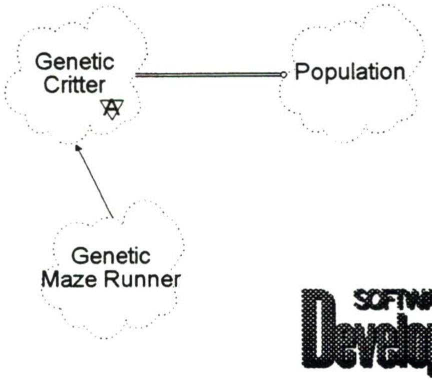
Encoding is All Important

- In light of schema theory, any scheme that is *reliant* on serial decoding of genotype devolves to very bad hill-climbing.
- **11 (or whatever) should have some inherent influence on overall fitness regardless of first two genes.
- #1 implementation problem

SOFTWARE Development

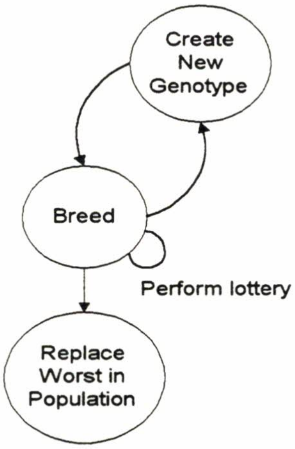

Create A Random Genotype
Birth
Translate genotype to phenotype
Growth
Evaluate phenotype

- So these aspects are different for each problem
- And therefore...
  - However, Population needs to interact with concrete subclasses.
  - Solution: Abstract Factory pattern (Gamma et al. 1995)


Genetic Critter
Population
Genetic Maze Runner

SOFTWARE Development

Population::Breed



- Normalize fitness
- Choose P1, P2
- Create new genotype
- Store genotype until all breeding is done
- Replace worst in population

SOFTWARE Development

Characteristics of GA

- Works with a population
- Stochastic
- Blind (operates on payoff only)
- Operates on the coding
- No known target, no error measure
- Very weak assumptions about payoff function
- No training period
- No central memory or bookkeeping
- Decentralized local control
- Implicit parallelism
- Fault-tolerant
- Audit trails
- Robust to environmental change
- Optimal allocation of trials
- Modeled after a known working system
- Computationally expensive but highly parallelizable

SOFTWARE Development

Simple Genetic System Source

- 1991
- Optimizes compile switches for BC++ version 1!

The 7 Major Steps

① **Design Encoding**
② `GeneticCritter::SetRandom()`
③ `GeneticCritter::Express()`
④ `Population::LotterySelect()`
⑤ `Population::Breed()`
⑥ `GeneticCritter::CreateFromSpecificGenotype()`
⑦ `Population::Display()`

Where to get more information

- Books
  - » Genetic Algorithms in Search, Optimization, and Machine Learning, David Goldberg, Addison-Wesey, Reading, MA, 1989
  - » Handbook of Genetic Algorithms, Lawrence Davis (ed.), Van Nostrand Rheinhold, New York, NY, 1991
  - » Genetic Programming I & II, John R. Koza, MIT Press, Cambridge, MA, 1991 & 1994
  - » Advances in Genetic Programming, Kenneth Kinnear Jr. (ed.), MIT Press, Cambridge, MA, 1994
- Magazines
  - » Artificial Life (MIT Press)
- Conferences
  - » A-Life
  - » Int'l Conference on Genetic Algorithms
  - » Int'l Congress on Computational Intelligence
  - » AAAI
- Online
  - »genetic-programming-request@cs.stanford.edu
  - »GA-List-Request@AIC.NRL.NAVY.MIL
  - »EP-List-Request@magenta.Me.Fau.Edu
  - »ftp.cc.utexas.edu /pub/genetic-programming
  - »http://alife.santafe.edu/
  - »http://gal4.ge.uiuc.edu/illigal.home.html
  - »CompuServe: GO AIEXPERT

For Source Code & Presentation Materials

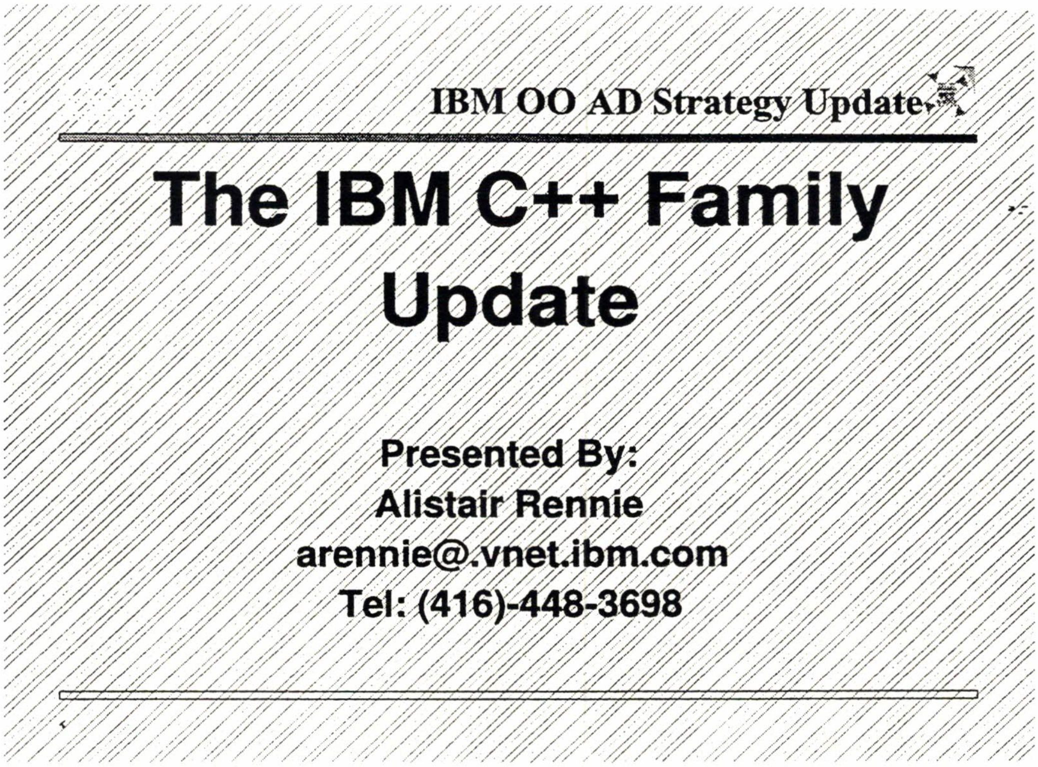
- Send email with subject "GA-REQUEST" to
  - » lobrien@mfi.com
- Or leave business card

PRO 0610

C Set ++ Overview

Alistair Rennie

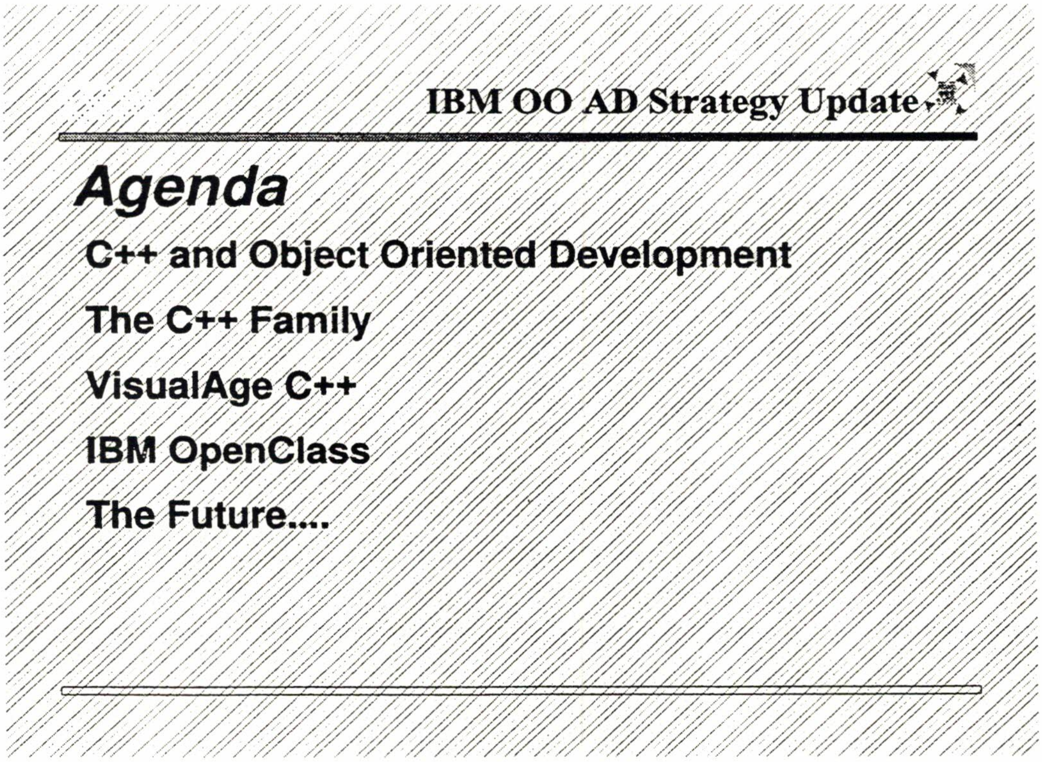
IBM



IBM OO AD Strategy Update

The IBM C++ Family Update

Presented By:
Alistair Rennie
arennie@.vnet.ibm.com
Tel: (416)-448-3698



IBM OO AD Strategy Update

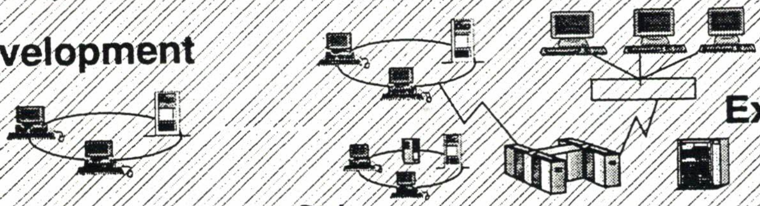
Agenda

- C++ and Object Oriented Development
- The C++ Family
- VisualAge C++
- IBM OpenClass
- The Future....

IBM OO AD Strategy Update

Today's Application Development Environment

Development



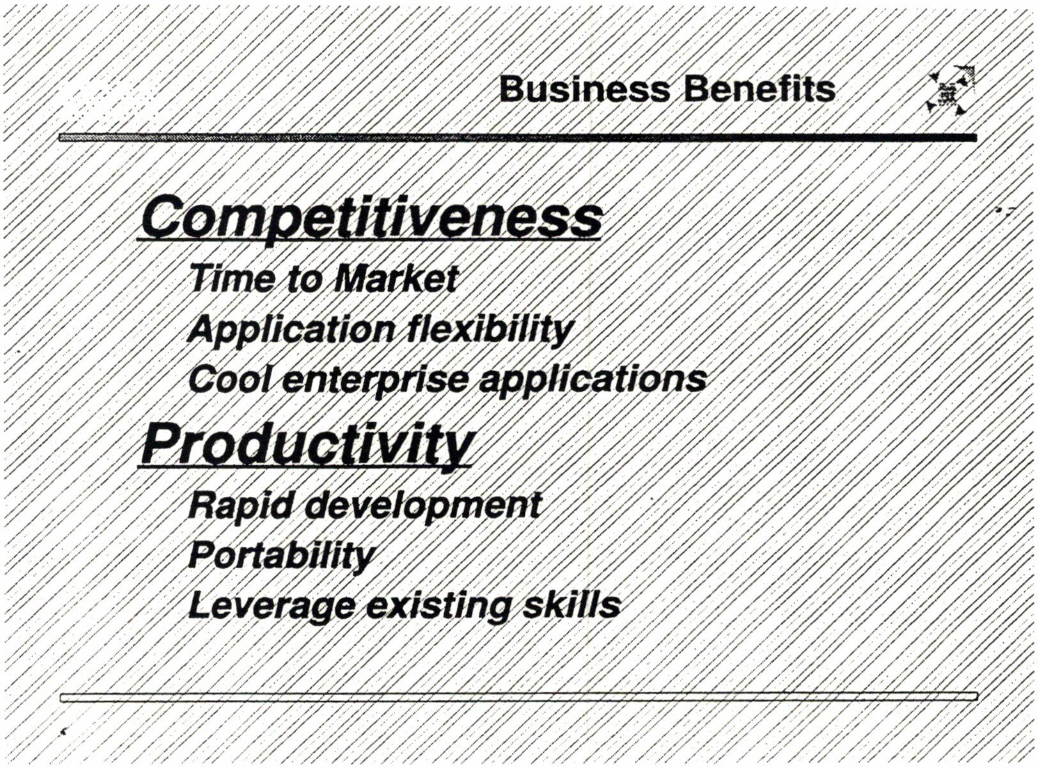
Execution

- Broad Scope of Development Styles
  - Enterprise, Workgroup, Individual
  - 3GL, 4GL, CASE, OO Tools
- Choice of Development Platforms
  - Intel, RISC, PowerPC (PPC)
- Pressures To Increase Productivity

- Broad Range of Applications
  - Mission Critical, OLTP, Decision Support, Personal..
- Client/Server
  - Access to Data Anywhere
  - Execution of Function Anywhere
- Heterogeneous Systems
  - OS/2, AIX, Windows, OS/400, MVS, UNIX

Customer AD Requirements

Executable Applications	• Are quickly modifiable • Execute quickly • Are H/W independent • Exploit client/server architectures
Individual Tools	• Develop distributed applications • Run quickly and on various platforms • Adjust to programmer's skill level • Supported by skilled staff • Technically state-of-the-art • Produce high quality applications • Low cost with flexible packaging
Development Environment	• Maximizes code re-use • Reduces cycle time • Provides testing facilities • Seamless tool integration • Facilitates change management
Investment Protection	• Skills and work are portable



Business Benefits

Competitiveness

- *Time to Market*
- *Application flexibility*
- *Cool enterprise applications*

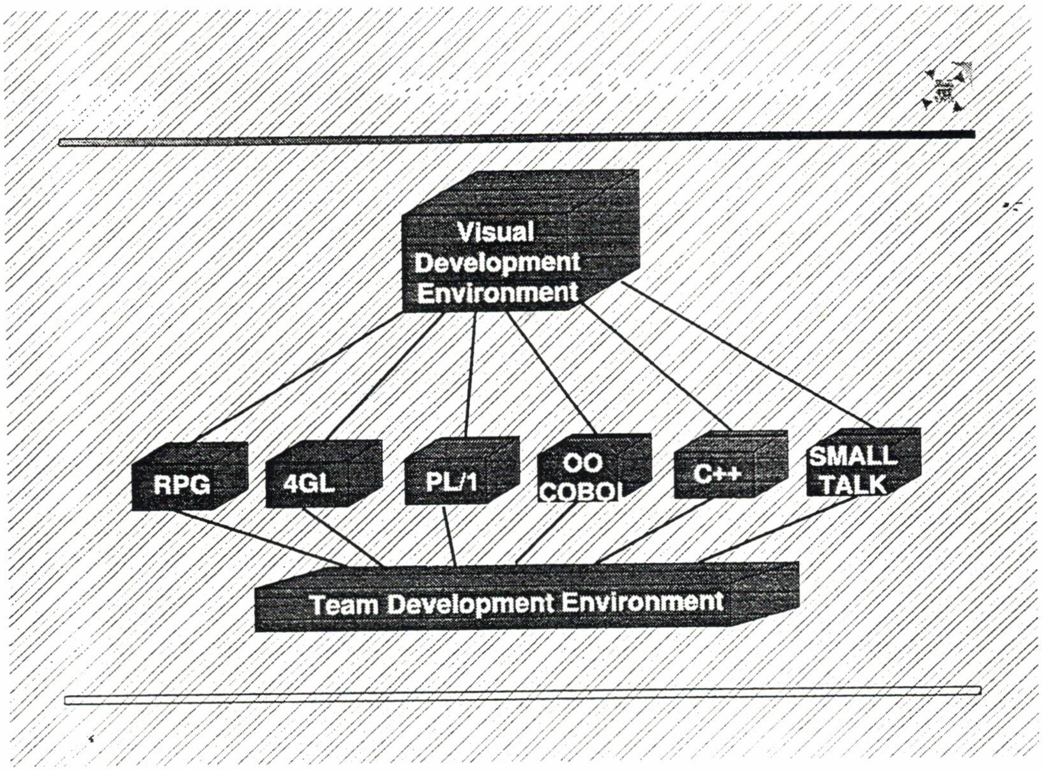
Productivity

- *Rapid development*
- *Portability*
- *Leverage existing skills*

Legacy Leverage

Protect Your Investment In:

- **Skills**
- **Code**
- **Systems**



Visual Development Environment

RPG

4GL

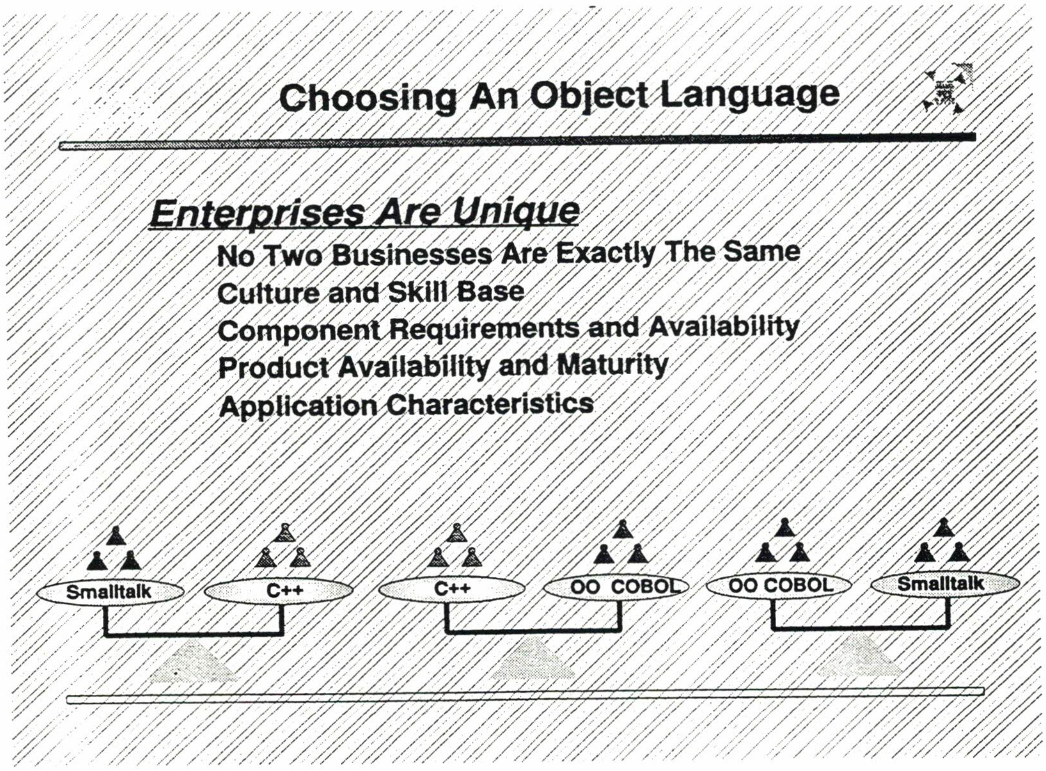
PL/1

OO COBOL

C++

SMALL TALK

Team Development Environment



Choosing An Object Language

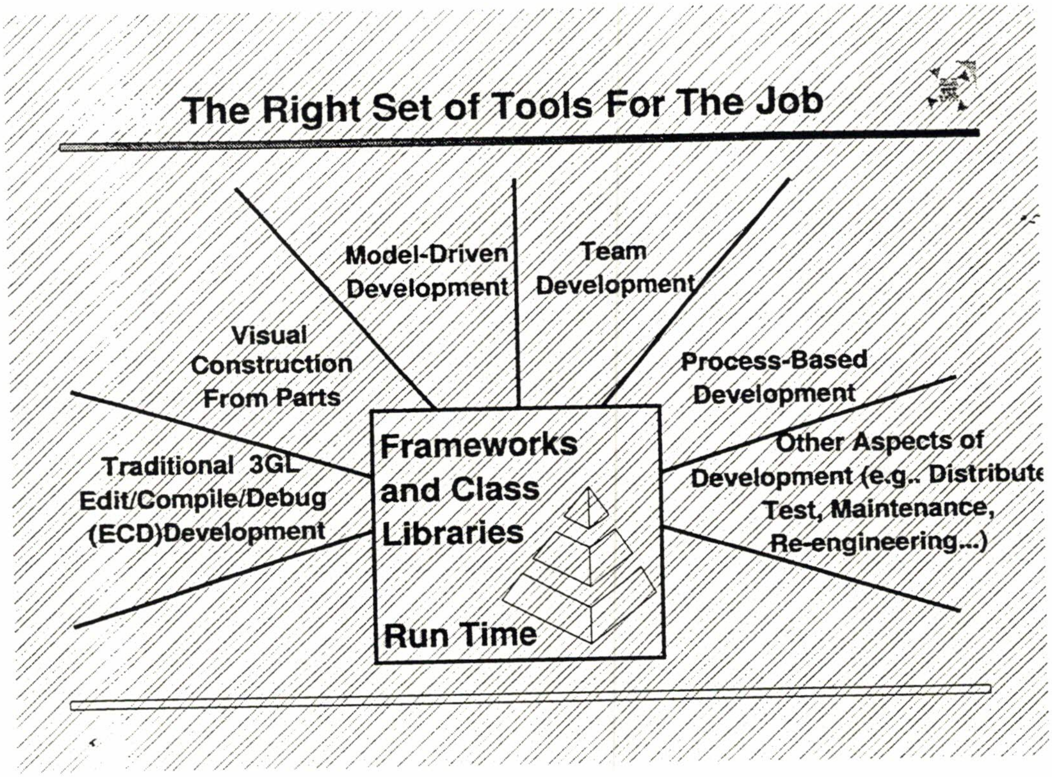
Enterprises Are Unique

- No Two Businesses Are Exactly The Same
- Culture and Skill Base
- Component Requirements and Availability
- Product Availability and Maturity
- Application Characteristics

Smalltalk — C++

C++ — OO COBOL

OO COBOL — Smalltalk



The Right Set of Tools For The Job

Model-Driven Development

Team Development

Visual Construction From Parts

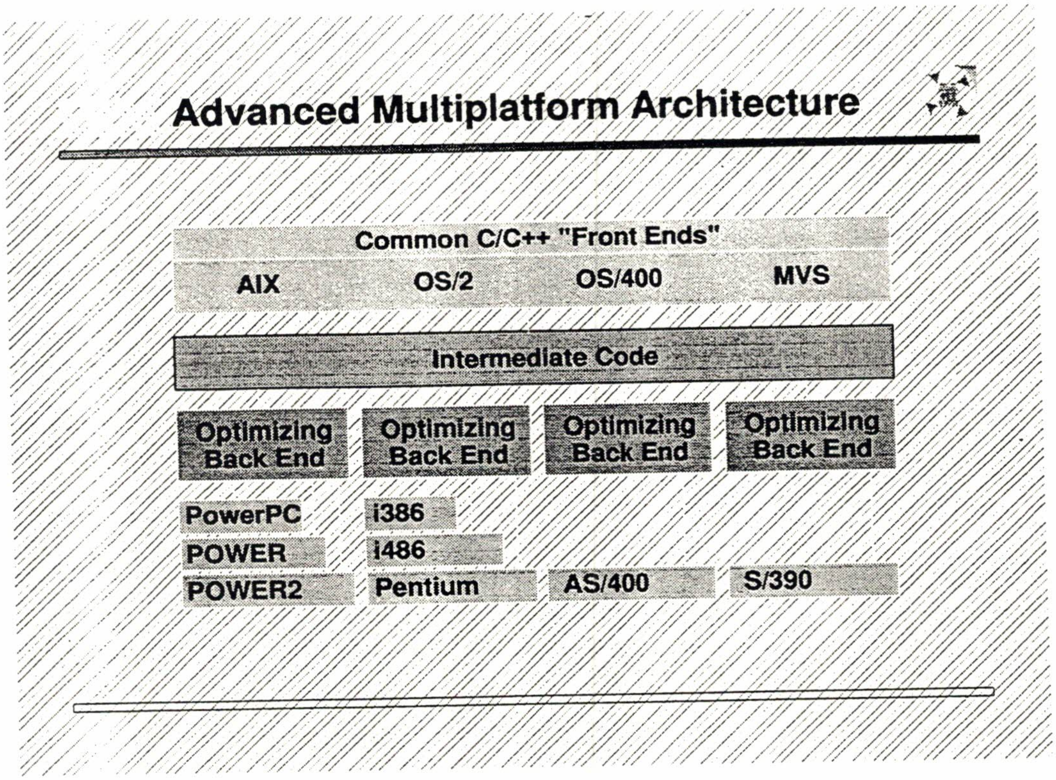
Process-Based Development

Traditional 3GL Edit/Compile/Debug (ECD)Development

Frameworks and Class Libraries

Run Time

Other Aspects of Development (e.g.. Distribute Test, Maintenance, Re-engineering...)



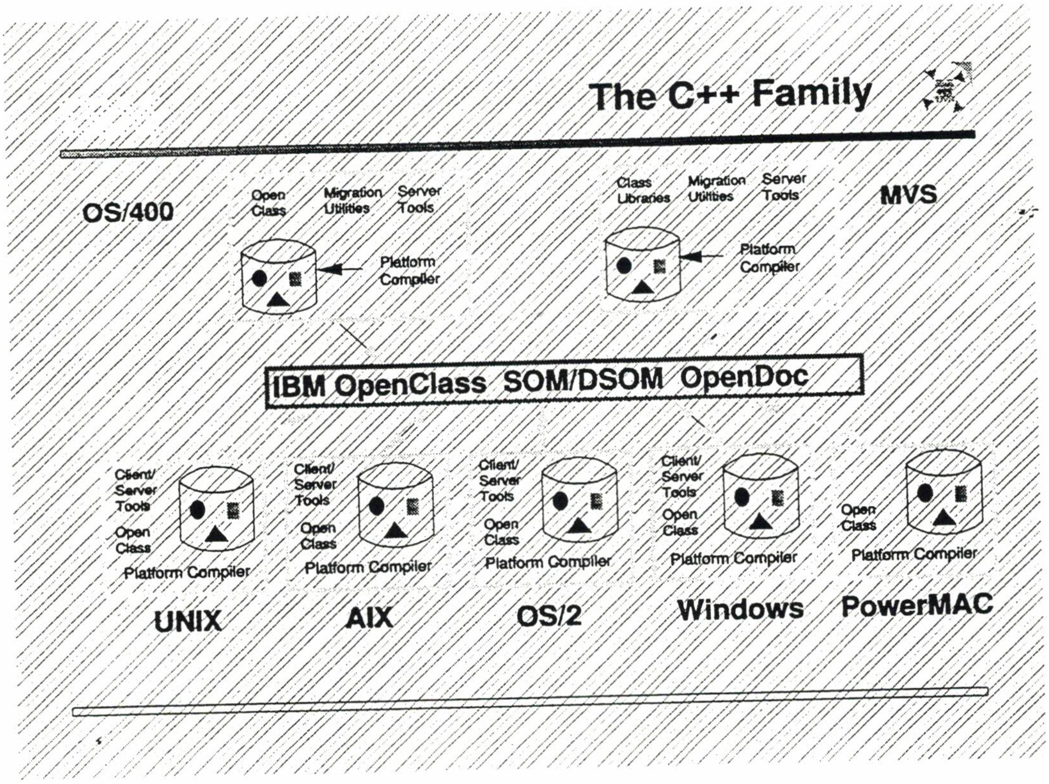
Advanced Multiplatform Architecture

Common C/C++ "Front Ends"

AIX OS/2 OS/400 MVS

Intermediate Code

Optimizing Back End	Optimizing Back End	Optimizing Back End	Optimizing Back End
PowerPC POWER POWER2	i386 i486 Pentium	AS/400	S/390



The C++ Family

OS/400

Open Class
Migration Utilities
Server Tools
Platform Compiler

Class Libraries
Migration Utilities
Server Tools
Platform Compiler

MVS

IBM OpenClass SOM/DSOM OpenDoc

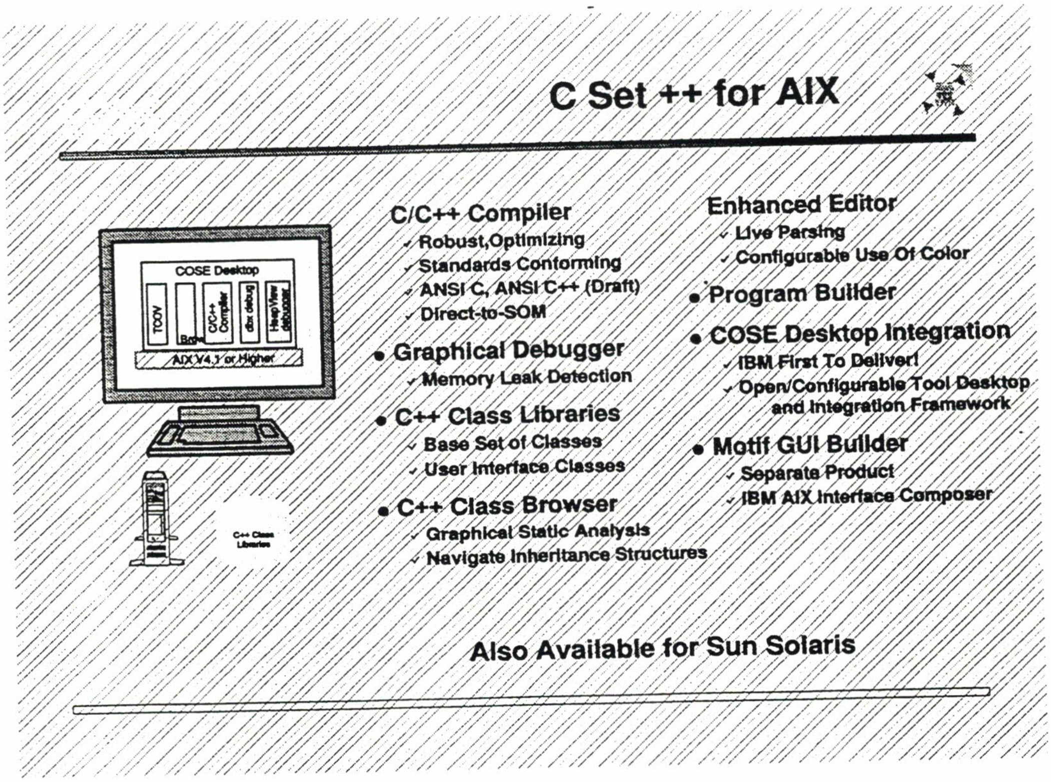
Client/ Server Tools
Open Class
Platform Compiler
UNIX

Client/ Server Tools
Open Class
Platform Compiler
AIX

Client/ Server Tools
Open Class
Platform Compiler
OS/2

Client/ Server Tools
Open Class
Platform Compiler
Windows

Open Class
Platform Compiler
PowerMAC



C Set ++ for AIX

COSE Desktop
TOOV
Bro
C/C++ Compiler
dbx debug
HeapView debugger
AIX V4.1 or Higher
C++ Class Libraries

C/C++ Compiler
- ✓ Robust,Optimizing
- ✓ Standards Conforming
- ✓ ANSI C, ANSI C++ (Draft)
- ✓ Direct-to-SOM

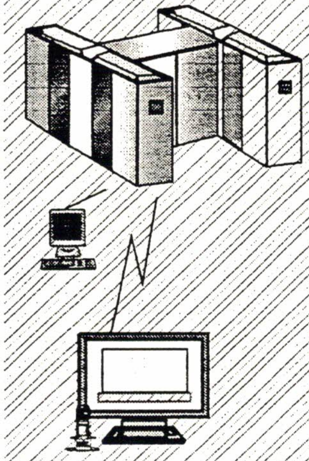
- **Graphical Debugger**
  - ✓ Memory Leak Detection
- **C++ Class Libraries**
  - ✓ Base Set of Classes
  - ✓ User Interface Classes
- **C++ Class Browser**
  - ✓ Graphical Static Analysis
  - ✓ Navigate Inheritance Structures

Enhanced Editor
- ✓ Live Parsing
- ✓ Configurable Use Of Color

- **Program Builder**
- **COSE Desktop Integration**
  - ✓ IBM First To Deliver!
  - ✓ Open/Configurable Tool Desktop and Integration Framework
- **Motif GUI Builder**
  - ✓ Separate Product
  - ✓ IBM AIX Interface Composer

Also Available for Sun Solaris

C Set ++ for MVS

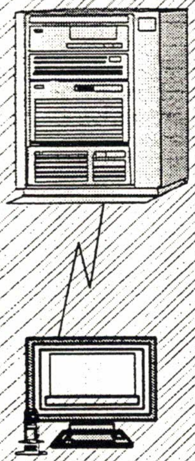


C/C++ Compiler
- ✓ Robust,Optimizing
- ✓ Standards Conforming
- ✓ ANSI C, ANSI C++ (Draft)
- ✓ Direct-to-SOM (SOM/MVS)
- ✓ Dynamic Link Library Support
- ✓ Consistent With C Set ++
- ✓ Based on Established C/370 Product

- **Mainframe Interactive Debugger**
  - ✓ Native To The Host
  - ✓ Supports CICS, IMS/ESA and DB2

C++ Class Libraries
- ✓ Base Set of Classes
  - IOStream
  - Complex
  - Collection
  - Application Support
- ✓ Data Access Class Generator
  - Work with DB2 without Awareness of SQL & DB2 Details

C Set ++ for OS/400



C/C++ Compiler
- ✓ Robust,Optimizing
- ✓ Standards Conforming
- ✓ ANSI C, ANSI C++ (Draft)
- ✓ Direct-to-SOM (SOM/MVS)
- ✓ Dynamic Link Library Support
- ✓ Consistent With C Set ++
- ✓ Based on Established C/400 Product

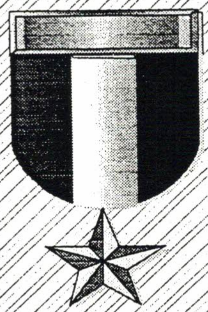
- **OS/2 Based Development Environment**
  - ✓ Cooperative compile

C++ Class Libraries
- ✓ Base Set of Classes
- ✓ Collection Classes
- ✓ Data Access Class Generator

A Powerful Vision

VisualAge C++: OS/2's Killer App"

Nicholas Petreley, InfoWorld May 8



Attendees Choice - Best New OT Product

Object World, Boston 95

Best New OT Development Product

Industry Judges, Object World, Boston 95

Construction from Parts

Parts Writers

- **Programmers**
- **Deep technical skills**
- **"Creator"**

Applications Builders

- **Have business perspective**
- **Know functional requirements**
- **"Assembler/Customizer"**

Visual & Non-Visual Parts

Visual Parts

Graphical User Interfaces
Multimedia
Forms / Data Entry

Non-Visual Parts

Logic and Data
Communications
Transactions

VisualAge C++ for OS/2

Compiler

Compiler
- Performance Improvements
- Pre-compiled Headers

Direct to SOM
- SOM Executables and IDL Debug and Browse Support

Linker
- 32-bit Rewrite Based On LINK386
- 2-4X Performance Improvement

Tools

Visual Builder

Data Access Builder
- Visual Schema Mapping and Data Access Tools

Performance Analyzer
- New UI

Editor
- Syntax-Directed Editor

Browser
- All new UI
- Source Code Browsing

Class Libraries

User Interface
- Toolbar, Multimedia, 2D

Collections

Data Classes

IDE

Workframe
- New Project Windows
- Tool Integration

Documentation
- Task Oriented - How do I?

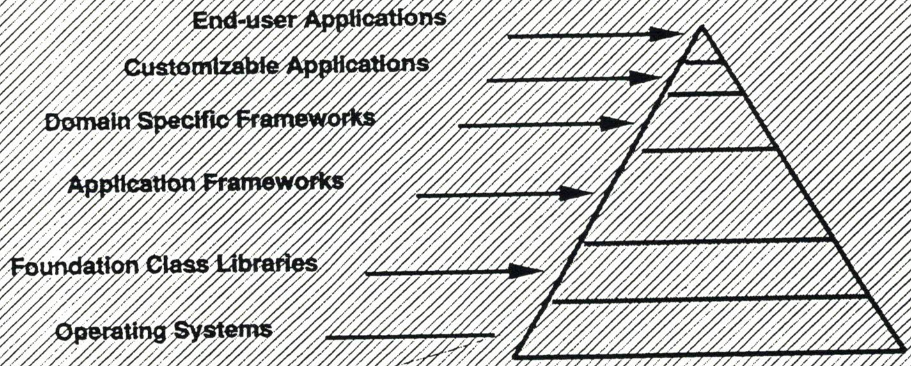
Install
- Single button install

IBM Open Class

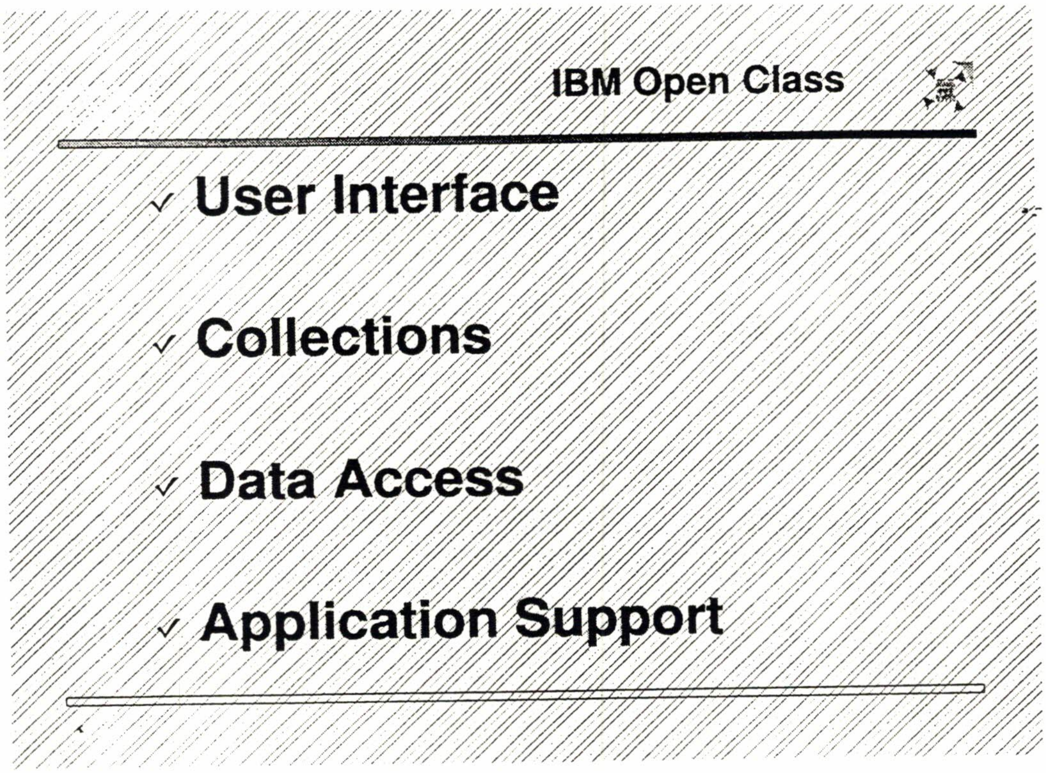
- Productivity From Reuse of Code and Design
- Abstraction of Middleware Complexity
- Reliable Portability Layer
- Superior Object-Oriented Design
- Evolve To Future Enhanced Designs

IBM Open Class

Class Libraries and Frameworks - What Are They?

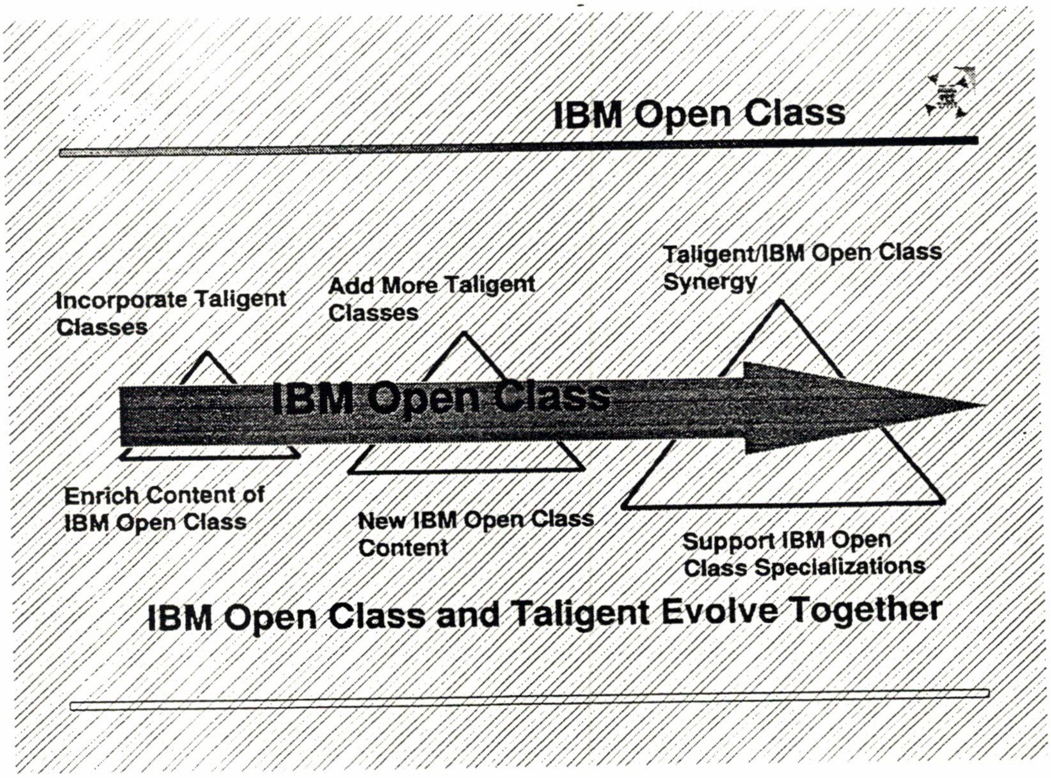

End-user Applications
Customizable Applications
Domain Specific Frameworks
Application Frameworks
Foundation Class Libraries
Operating Systems

Programming at Higher Levels of Abstraction Reduces Development Complexity



IBM Open Class

- ✓ User Interface
- ✓ Collections
- ✓ Data Access
- ✓ Application Support



IBM Open Class

Incorporate Taligent Classes
Add More Taligent Classes
Taligent/IBM Open Class Synergy
IBM Open Class
Enrich Content of IBM Open Class
New IBM Open Class Content
Support IBM Open Class Specializations

IBM Open Class and Taligent Evolve Together

The Future !

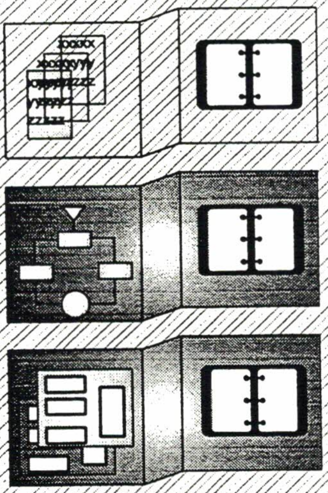
More IBM Open Class Content

OO Modelling Tools

Integrated Team Support

Framework Based Environments

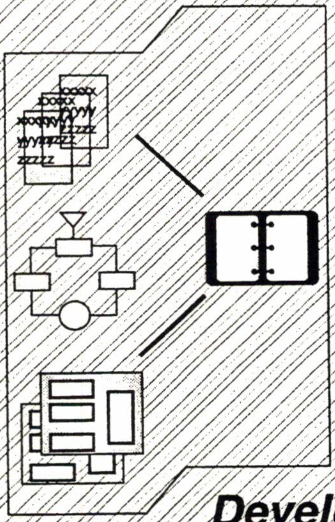
Model-Driven Development



Current "CASE" Tools

- **Model and Code Go Out of Synch**
- **Re-engineering of Dynamic Behavior Is N Possible**
- **Misunderstandings in Models Is Detected Much Later**

Object Modeling Technology



Ideal Modeling Tool

- Model Becomes Prototype
- Early Validation With Continuous User Involvement
- Generates Complete Code, Not Just Skeletons
- Live Documentation

Develop At Business Object Level

Other Presentations

- Welcome to C Set ++ (VisualAge) V3
- C Set ++ Class Library
- Visual Builder for C Set ++
- C Set ++ and SOM
- Overview of C Set ++ for AIX
- C++:Cross Platform Solution (MVS)
- C Set ++ for OS/400
- From Bugs to Blazing

SYS 0606

DB2/2 Performance and Tuning

Rick Weaver
IBM

Agenda

- General Tuning Concepts
- Database Design and Performance
- SQL
- Parameters, Parameters, and more Parameters
- Tools to Help the Tuning Effort
- Demo

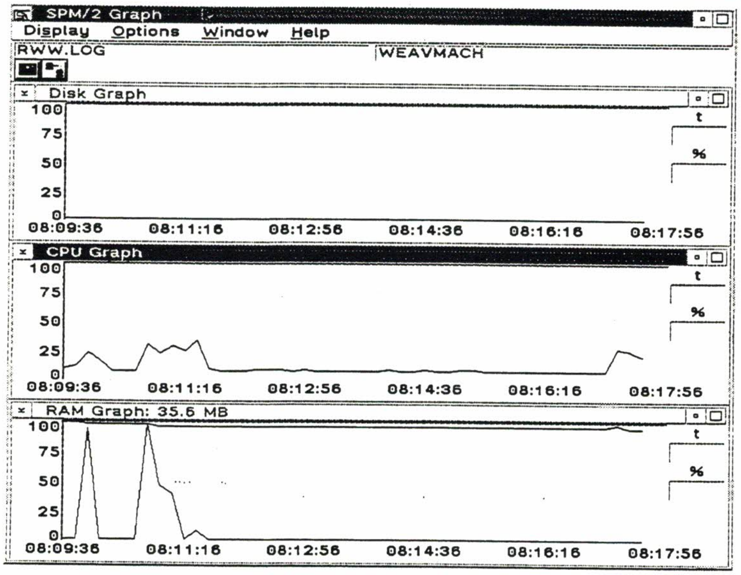
General Tuning - Why Tune?

- Installed defaults often inappropriate
  - Chosen for minimum memory requirements.
  - All applications are different.
- Meet performance objectives
  - End user expectations
  - Service Level Agreements
  - Strategic application SQL events
  - Application startup events

What Affects Performance?

- System Performance
  - CPU
  - DISK
  - Memory (Working Set and Allocated memory)
- Database Design
  - Physical &Logical Database Design
  - Table Adminstration
- SQL
- Parameters
  - Database System and Database Parameters
  - OS/2 System Level
- Application Design

System Performance

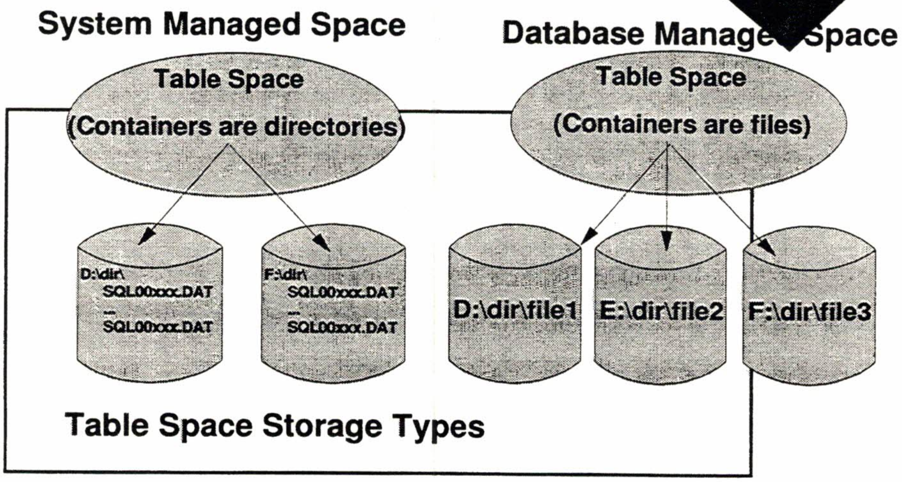

SPM/2 Graph
Display Options Window Help
RWW.LOG
WEAVMACH
Disk Graph
100
75
50
25
0
08:09:36
08:11:16
08:12:56
08:14:36
08:16:16
08:17:56
t
%
CPU Graph
100
75
50
25
0
08:09:36
08:11:16
08:12:56
08:14:36
08:16:16
08:17:56
t
%
RAM Graph: 35.6 MB
100
75
50
25
0
08:09:36
08:11:16
08:12:56
08:14:36
08:16:16
08:17:56
t
%

Agenda - Database Design

- Physical Design
  - Containers and Tablespaces *
  - Logs
- Logical Design
  - Tables and Indices
  - Prefetching *
  - LOBS *
- Table Administration
  - Reorgs
  - Runstats

*** Only applicable for DB2/2 Version 2.1**

Physical Design - Tablespaces


System Managed Space
Database Manage Space
Table Space
(Containers are directories)
Table Space
(Containers are files)
D:\dir\
SQL00xxx.DAT
...
SQL00xxx.DAT
F:\dir\
SQL00xxx.DAT
...
SQL00xxx.DAT
D:\dir\file1
E:\dir\file2
F:\dir\file3
Table Space Storage Types

Physical Design - Tablespaces

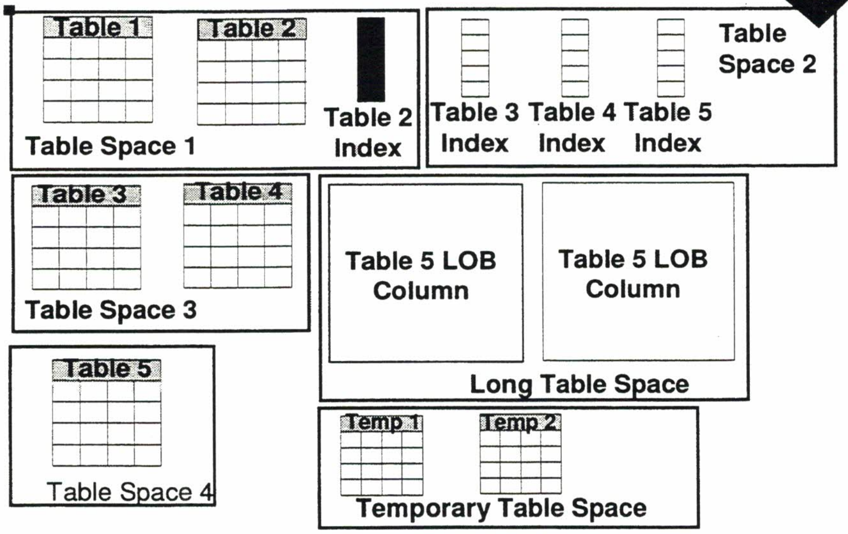


Table 1
Table 2
Table 2 Index
Table Space 1

Table 3 Index
Table 4 Index
Table 5 Index
Table Space 2

Table 3
Table 4
Table Space 3

Table 5 LOB Column
Table 5 LOB Column
Long Table Space

Table 5
Table Space 4

Temp 1
Temp 2
Temporary Table Space

Benefits of Tablespaces

- Database size can span logical devices.
- Reduction of Disk I/O contention.
  - Only if Tables/Indices are on separate physical devices.
  - Disk Arrays limit the flexibility of placement of tablespaces.
- Recoverability

DB2/2 Transaction Logs

- First, look at transaction types:
  - Heavy Insert/update,delete based transactions?
    - Separate Transaction logs on a separate physical disk.
    - Database configuration parameter.
  - Heavy DSS use?
    - Not a high return if transaction logs are moved.
- Forward Recovery Enabled?
  - Choice of how online archive transaction logs are moved
  - is critical to how forward recovery impacts production
  - usage.

Logical Design - Tables

- Tables
  - Normalization vs. Denormalization
    - How many joins typically occur?
    - OLTP - Avoid more than three table join.
  - DB2/2 1.x has a join limit of 14 tables
    - Not that you would WANT a 14 table join occuring in
    - production anyway!
  - Storing data in LONG VARCHAR's or LOBS causes
  - additional I/O. (Stored outside of the table)
  - VARCHAR vs. CHAR

Logical Design - Indices

- Proper Index definition critical for performance.
  - RI - Foreign keys should be indexed.
  - Common search criteria on WHERE clause should be
  - indexed.
- Use of EXPLAIN is critical for analyzing whether proper
- indexing has been done.
  - Explain has changed drastically with Version 2.1.
    - Visual Explain as well as the EXPLAIN statement.
- Always a tradeoff between SELECT performance and
- INSERT, UPDATE, and DELETE performance.
- Try to create index columns in the order of decreasing
- cardinality.

Prefetching (V 2.1)

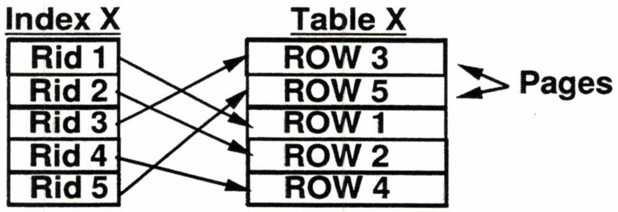
- Data pages are moved into the buffer pool in anticipation of their use.
- # of prefetched pages determined by PREFETCHSIZE.
- Type types of prefetching:
  - Sequential prefetch:
    - Table Scans are automatically use sequential prefetch.
    - Sequential detection for other queries can be enabled too.
  - List prefetch:
    - If data pages are not sequential, DB2/2 can sort the
    - RID's to satisfy the query, then return the data.
    - Increases number of rows retrieved in a single access.

LOBs and Performance

- LOB's are large objects (up to 2GB per LOB) stored in the database.
  - Use NOT LOGGED to avoid additional transaction LOG I/O.
    - Should use a separate table space for recovery.
  - COMPACT option reduces disk size of LOB storage.
    - Performance penalty if LOB is updated with a larger size.

Table Administration

- Reorgs/Runstats critical for performance on dynamic tables
  - Rows becomes fragmented in table(s):



Index X

Rid 1
Rid 2
Rid 3
Rid 4
Rid 5

Table X

ROW 3
ROW 5
ROW 1
ROW 2
ROW 4

Pages

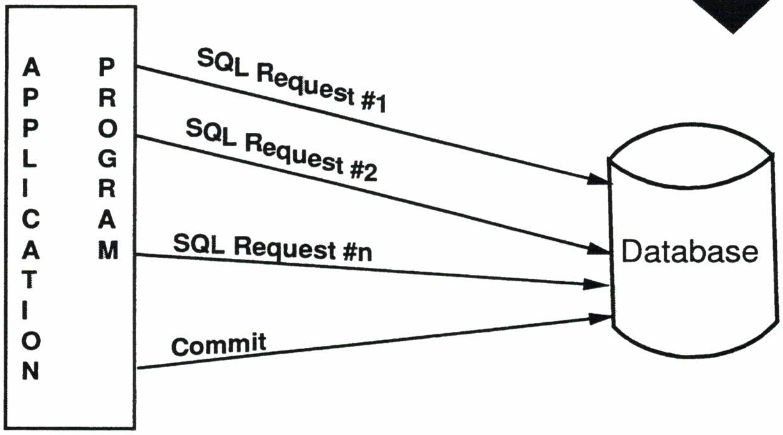
- REORGCHK should be used to determine whether REORG
- should be run.
- Always run RUNSTATS after REORG.
- Should rebind all static SQL packages.

Agenda - SQL

- Compound SQL *
- DARI
- Triggers *
- Optimizer enhancements *
- Dynamic vs. Static SQL

*** Only applicable for DB2/2 Version 2.1**

Compound SQL and DARI


APPLICATION PROGRAM
SQL Request #1
SQL Request #2
SQL Request #n
Commit
Database

Anything wrong with this scenario?

Compound SQL

- Combines multiple SQL statments into one executable block.
- Reduces network traffic->Increases performance.
- Only SQL statements can make up a compound SQL block.
- DDCS/2 only supports NOT ATOMIC.

```
BEGIN COMPOUND ATOMIC STATIC
   STOP AFTER FIRST :number STATEMENTS
   SQL statement #1; SQL statement #2;
   SQL statement #n; COMMIT;
END COMPOUND
```

DARI

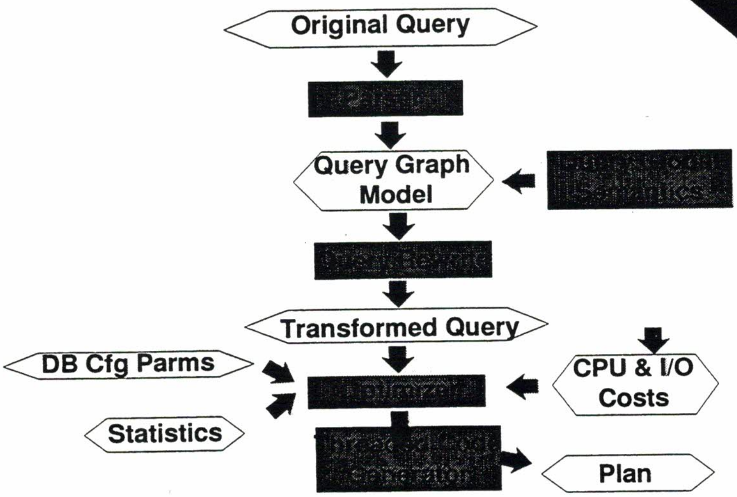
- DARI is coded in a user-written DLL (in a 3-GL).
- DLL is stored on the server.
- More flexible than compound SQL but more difficult code
- Client:
  - Builds input SQLDA
  - Issues sqleproc with input SQLDA & DLLname/funcname.
- Server:
  - Receives input SQLDA
  - Runs DLL/function
  - Builds output SQLDA
  - Passes output SQLDA and SQLCA to client

Triggers

- Defines a set of actions which are executed by an
- update operation on a base table.
- Update -> Insert, Update, and Delete
- Triggers can perform SQL or call a UDF.
- Can reduce application complexity -> Applications perform faster because enforcement of business rules are done at the server.

```
CREATE TRIGGER new_hire
   AFTER INSERT ON employee
   FOR EACH ROW
   update company_stats
   set number_employees=number_employees + 1
```

Optimizer Enhancements


Original Query
Parser
Query Graph Model
Semantics
Query Rewrite
Transformed Query
DB Cfg Parms
Statistics
Optimizer
CPU & I/O Costs
Threaded Code Generator
Plan

SQL Compile Process

Query Rewrite

- Tranforms the original query into a more optimizable query.
- In general, optimizer attempts to tranform queries from a row-at-a-time to set-oriented execution.
- Some of the transformations include:
  - Redundant join elimination
  - Convert OR to IN
  - Convert IN to join
  - View merging
  - Distinct elimination
- In version 1.x, the SQL developer had to do all the above.

Optimization (Version 2.1)

- Cost based optimizer looks at:
  - CPU costs (uses CPU speed as a statistic)
    - Average elapsed time in milliseconds per CPU instruction.
  - I/O costs:
    - Overhead of disk controller.
    - Seek and latency times
    - Time in milliseconds to read a 4K page.
  - RUNSTATS data:
    - Distribution statistics for column values.
    - Pages accesses required during an index scan.

Static vs. Dynamic SQL

- Static SQL is compiled BEFORE execution of a statement.
- Dynamic SQL is compiled DURING execution of a
- statement.
- Some of the optimizer enhancements will result in longer
- compile time of the SQL.
- DB2/2 allows you to specify a query optimization level for a
- plan:

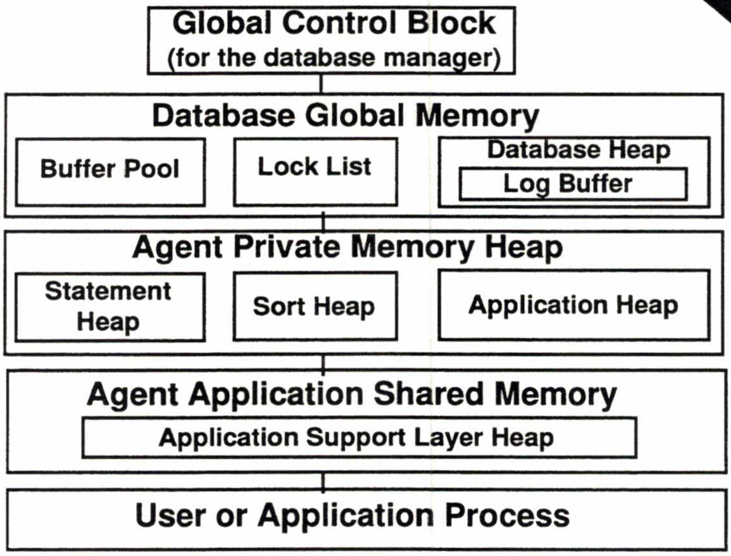
Optimization Level	Typical Usage
Minimal	Only when low overhead is priority, very simple queries on well indexed tables.
DB2 V1	General purpose use in a CPU or memory constrained system.
DB2/MVS	Suitable for broad range of applications, has fewer options than default.
DB2 V2	Mixed environment for both simple and complex transactions.
Maximum	Only for complex or long-running queries.

Understanding DB2/2 Parameters

- How DB2/2 uses memory
- DB2/2 Database Parameters
- DB2/2 System Parameters

Please see the Administirator Guide for an exhaustive list of all database parameters.

Memory usage by DB2/2



Global Control Block
(for the database manager)

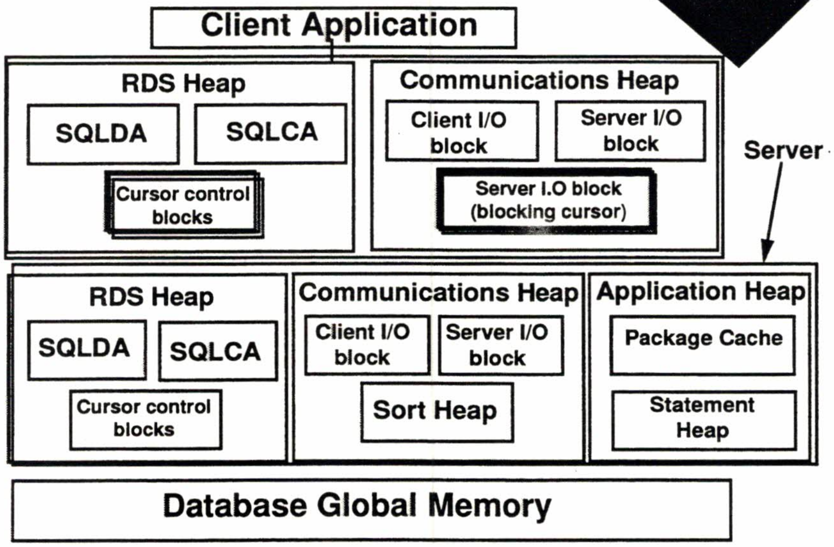
Database Global Memory
Buffer Pool
Lock List
Database Heap
Log Buffer

Agent Private Memory Heap
Statement Heap
Sort Heap
Application Heap

Agent Application Shared Memory
Application Support Layer Heap

User or Application Process

Memory usage by DB2/2 - Remote



Client Application

RDS Heap
SQLDA
SQLCA
Cursor control blocks

Communications Heap
Client I/O block
Server I/O block
Server I.O block (blocking cursor)

Server

RDS Heap
SQLDA
SQLCA
Cursor control blocks

Communications Heap
Client I/O block
Server I/O block
Sort Heap

Application Heap
Package Cache
Statement Heap

Database Global Memory

Database Parameters

- BUFFPAGE
  - Most important parm of them all!
  - Used to:
    - hold active pages in memory
    - avoid disk I/O
  - Pages read in as needed; LRU algorithm for writing out.
  - 1 per database: used by all attached processes.
  - NEVER set to the largest size without benchmarked
  - justification!
  - Range: [2*maxappls - 524288]

Database Parameters

- Average Active Applications (AVG_APPLS)
  - Used by SQL optimizer in multi-user environment.
  - Parm influences the optimizer to choose an access plan that will be more conservative in BUFFPAGE usage.
  - Useful for databases with multiple complex queries.
  - Rebind all static SQL after making changes to parm.
  - Range: [1 - maxappls]

Database Parameters

- Log Buffer Size (logbufsz)
  - Buffer for log records before they are written to disk.
  - Physical write to log occurs:
    - A transaction issues a commit.
    - A group of transactions commit as governed by mincommit.
    - The log buffer is full.
  - Logbufsize>= mincommit*(log space used per txn)
- Number of Commits to Group (mincommit)
  - Delays writing log records until a minimum # of commits.
  - have been issued.
  - Good if heavy log activity is occurring.

Database Parameters

- Changed Pages Threshold (chngpgs_thresh)
  - Defines the % of changed pages after which
  - asynchronous page cleaners are started.
  - Page cleaners write changed buffer pages to disk before the space is needed by other transactions.
  - Range: [5-80]
- Number of Asynchronous Page Cleaners (num_iocleaners)
  - Set to zero for any read-only databases.
  - Very important for updateable databases.
  - Set to # of physical storage devices for a database.
  - Range: [0-16]

Database Parameters

- Number of I/O Servers (num_ioservers)
  - Performs prefetch I/O.
  - Non-prefetch I/O still occur within the database agent.
  - Recommendation: set to 1 or 2 more than the number of physical devices on which the database resides.
  - Minimal overhead associated with each I/O server.
  - Range: [1-255]
- Sequential Detect (seqdetect)
  - Monitors I/O to see if I/O prefetching can occur.
  - Sequential detection still will occur on tables scans, list prefetches, etc.
  - Range [Yes,No]

Database Parameters

- LOCKLIST
  - Maximum storage for the lock list.
  - Locks use 32 or 64 bytes of the locklist.
  - One lock list per database.
  - Range [4-250]
- MAXLOCKS
  - A percentage of the locklist size.
  - If exceeded by one application, escalation occurs.
  - Range [1-100]
- DLCHKTIME
  - Time interval for checking for deadlocks.

Database Parameters

- SORTHEAP
  - A work area for:
    - Data being sorted (order by, group by, distinct, etc.)
    - Internal control structures used during the sort.
  - Very important to keep in memory as much as possible.
- SHEAPTHRES
  - A threshold, not an allocation.
  - Controls amount of SORTHEAP used by sorts.
  - If this is exceeded, the likelihood of a disk I/O sort
  - increases.

Tuning Tools

- Benchmarking
- SPM/2
- Explain
- Performance Monitor *

*** Only applicable for DB2/2 Version 2.1**

Benchmarking

- Create a special program containing ***selected*** SQL statement(s)
- Embed timing statements around the SQL
- Run a baseline (control) test
  - default tuning parms
- Select a parm and raise it

Better timings:

- ➢ raise same parm
- ➢ run test again
- ➢ look for elbow of curve

Worse/ No Change:

- ➢ return to previous value
- ➢ try another parm

SPM/2

- OS/2 Resource Monitor
  - CPU Utilization
    - System
    - Process/Thread
  - Memory Utilization
    - Memory Usage
    - Working Set *
  - Disk Utilization
    - Physical Design
    - Logical Design

Explain

- Explain for Version 1.x
  - Displays textual report of access paths for all SQL in static SQL packages.
  - Dynamic SQL has to be traced, then ported into a static SQL program.
- Explain has changed for Version 2.1
  - Explain Table.
  - Graphical tool to display each step of query resolution.
  - Packages should be bound as explainable.
  - Total cost of query shown in timerons.
  - Dynamic SQL can be explained using explain statement.
- Both are critical for performance tuning.

Performance Monitoring for DB2/

- New with DB2/2 Version 2.1!
- Performance monitor GUI frontend to monitoring API's
  - Snapshot monitoring:
  - (Returns information for a specific point in time)
    - Sorting information
    - Lock information
    - Table activity
    - Buffer pool activity information
    - Unit of work Information
    - SQL statement information

Performance Monitoring for DB2/

- Event Monitoring
- (Returns information over a period of time)
  - Database
  - Connections
  - Tables
  - Statements
  - Transactions
  - Deadlock
  - Tablespaces

SYS 0607

NetSP Secured Network Gateway

Peter Crotty
IBM

IBM NetSP Secure Network Gateway (Internet Firewall)

Peter J. Crotty
Senior Marketing Rep, NetSP Firewall

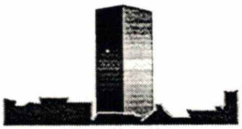
P. O. Box 12195
B44A/B501
RTP, N. C. 27709

Phone: **(919) 254-5074**
X.400: **US, IBMX400, IBMMAIL, CROTTY, PETER**
Internet: **peter_crotty@vnet.ibm.com**

Copyright 1995 IBM Corporation

Changing Complexion of the Internet

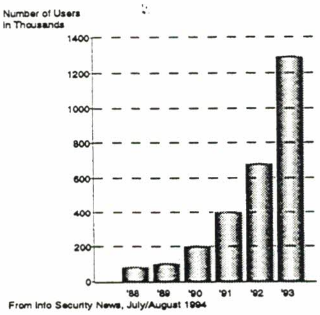
- **The Old Internet -**
  - E:Mail (SMTP) and file transfer (FTP) for the academic, research community



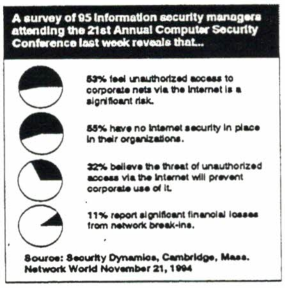
- **The New Internet -**
  - World Wide Web (WWW) servers and Mosaic Clients, distributing commercial information

Copyright 1995 IBM Corporation

The Threat



From Info Security News, July/August 1994



A survey of 95 information security managers attending the 21st Annual Computer Security Conference last week reveals that...

53% feel unauthorized access to corporate nets via the Internet is a significant risk.

55% have no Internet security in place in their organizations.

32% believe the threat of unauthorized access via the Internet will prevent corporate use of it.

11% report significant financial losses from network break-ins.

Source: Security Dynamics, Cambridge, Mass.
Network World November 21, 1994

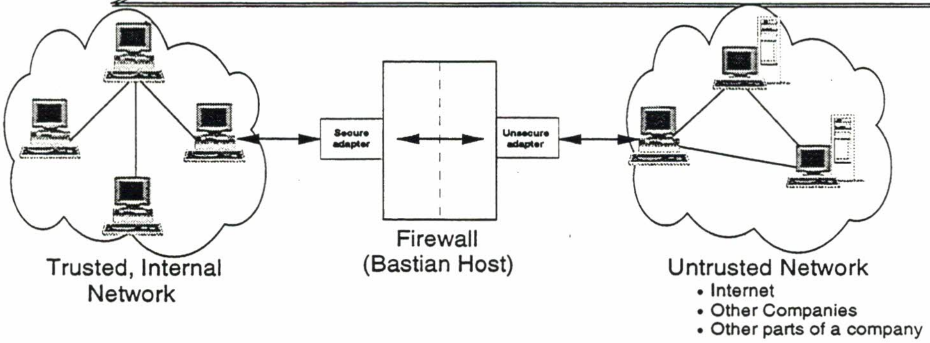
Security Awareness Down

"One-third of the 1,271 users surveyed had experienced financial losses due to security breaches in their networks during the past two years...
... 17 respondents said the breaches cost their companies $1 million"

Communications Week, 12/5/94

- **Explosive Growth of the Internet**
  - Number of connected hosts and networks
  - Amount of commercial usage
- **Makes the Internet . . .**
  - Desirable place to do business
  - Attractive place to steal from business

What is a Firewall?


Secure adapter
Unsecure adapter
Trusted, Internal Network
Firewall (Bastian Host)
Untrusted Network
• Internet
• Other Companies
• Other parts of a company

- **Prevents unauthorized traffic in and out of a secure network**
- **Address only TCP/IP accessible networks**
- **Dedicated machine; one per network**

Good Firewall Design

- Anything not explicitly permitted should default to denied
- Increasing complexity leads to bugs leads to "opportunities"
- Physically secure
- Provide extensive logging
- Turn off known problems and non essential daemons - 'R*' commands
- Dedicated machine, don't run other apps

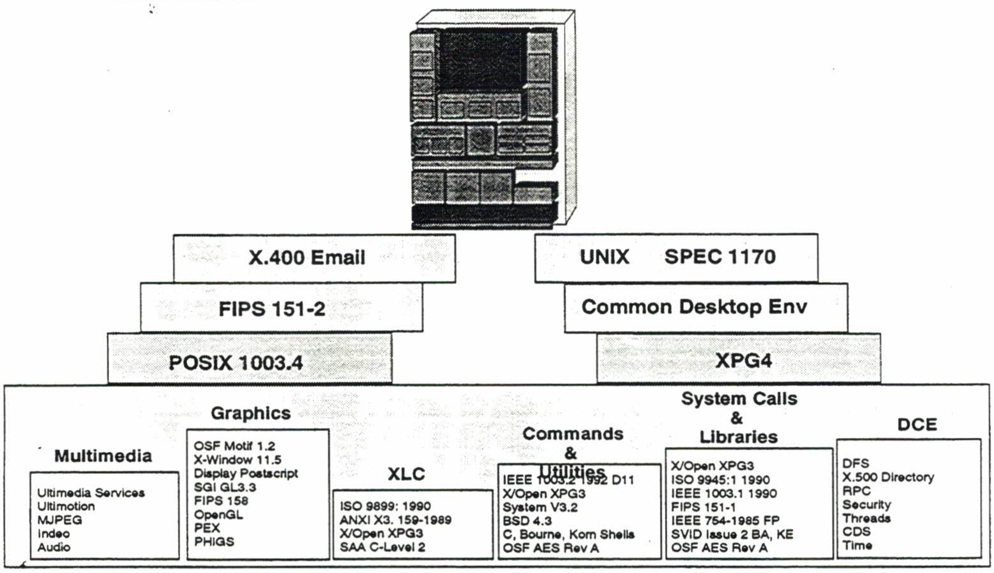
Copyright 1995 IBM Corporation

Customer Platform Requirements

- Commercial, off-the-shelf technology
- High performance hardware
- Open systems, standards based operating environment
- Highly scalable platform

Copyright 1995 IBM Corporation

AIX Open Systems Leadership



X.400 Email

FIPS 151-2

POSIX 1003.4

UNIX SPEC 1170

Common Desktop Env

XPG4

Multimedia
- Ultimedia Services
- Ultimotion
- MJPEG
- Indeo
- Audio

Graphics
- OSF Motif 1.2
- X-Window 11.5
- Display Postscript
- SGI GL3.3
- FIPS 158
- OpenGL
- PEX
- PHIGS

XLC
- ISO 9899: 1990
- ANXI X3. 159-1989
- X/Open XPG3
- SAA C-Level 2

Commands & Utilities
- IEEE 1003.2 1992 D11
- X/Open XPG3
- System V3.2
- BSD 4.3
- C, Bourne, Korn Shells
- OSF AES Rev A

System Calls & Libraries
- X/Open XPG3
- ISO 9945:1 1990
- IEEE 1003.1 1990
- FIPS 151-1
- IEEE 754-1985 FP
- SVID Issue 2 BA, KE
- OSF AES Rev A

DCE
- DFS
- X.500 Directory
- RPC
- Security
- Threads
- CDS
- Time

Copyright 1995 IBM Corporation

AIX: Powerful, Scalable

One Implementation - Customized Environment

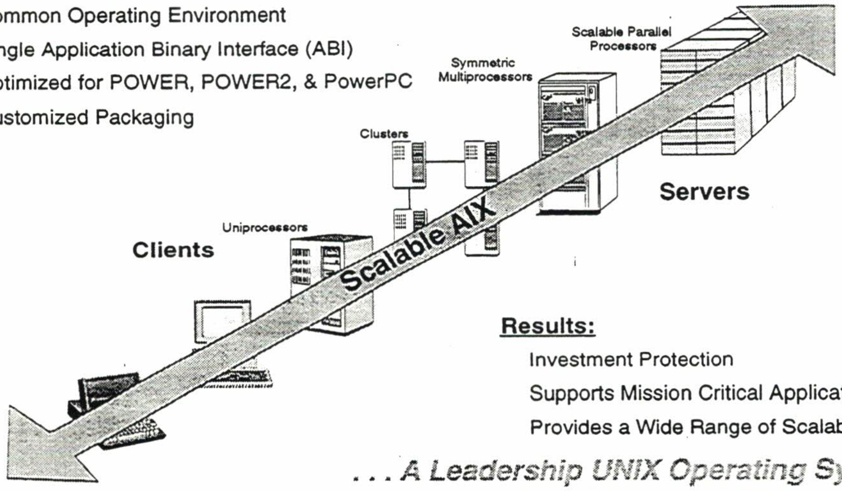
Objectives:

Common Operating Environment

Single Application Binary Interface (ABI)

Optimized for POWER, POWER2, & PowerPC

Customized Packaging



Results:

Investment Protection

Supports Mission Critical Applications

Provides a Wide Range of Scalability

. . . A Leadership UNIX Operating System

Copyright 1995 IBM Corporation

Customer Firewall Requirements

- **Ease of configuration and administration**

- **Flexible, extensive choices of firewalling technology**

- **Complete logging**

- **Alarms to network administrators**

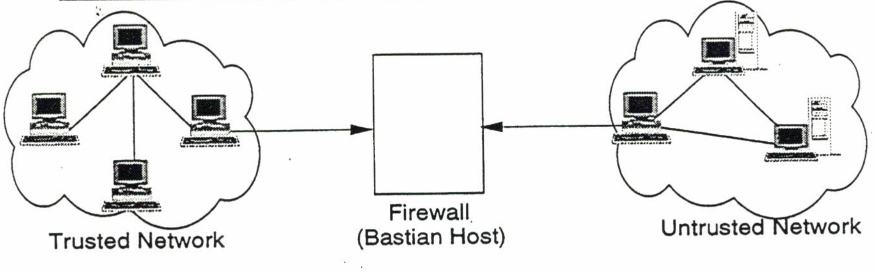
SNG Key Technologies

1. **Filter**
2. **Proxy**
3. **Socks**
4. **DNS**
5. **Mail Handling**
6. **Authentication**

- **Multiple technologies needed to solve the problem**

- **SNG is based on IBM's technology used for seven years to protect internal IBM networks**

Filter


Trusted Network
Firewall
(Bastian Host)
Untrusted Network

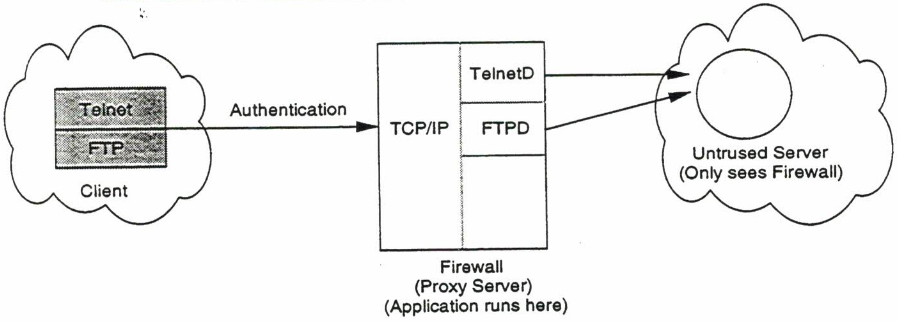
- **Filter Rules defined on Firewall**
  - Either permit or deny
- **Filter incoming/outgoing traffic based on TCP/IP info**
  - Source/Destination IP address
  - Source/Destination port
  - Local vs. routed traffic
  - IP protocol (UDP, ICMP, TCP, TCP w/acknowledgement)
- **Limited Applicability**
  - Some services use random points
  - UDP address can be altered
  - Best if used with other technologies (Proxy, Socks)

Networking Products

Copyright 1995 IBM Corporation

Firewall 11

Proxy or Application Gateway


Telnet
FTP
Client
Authentication
TCP/IP
TelnetD
FTPD
Untrused Server
(Only sees Firewall)
Firewall
(Proxy Server)
(Application runs here)

TCP/IP

- Uses existing applications as is
- Good for efficient applications like Telnet and FTP
- Inbound Telnet with Smartcard

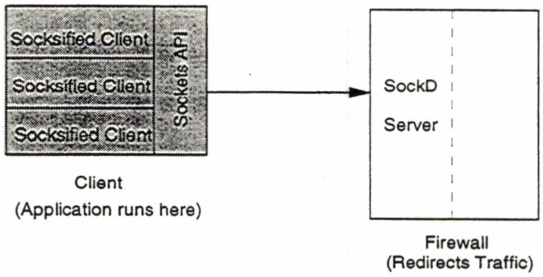
Supported Applications

- Telnet, FTP
- Gopher, Mosaic

Networking Products

Copyright 1995 IBM Corporation

Firewall 12


Socksified Client
Socksified Client
Socksified Client
Sockets API
Client
(Application runs here)
SockD
Server
Firewall
(Redirects Traffic)

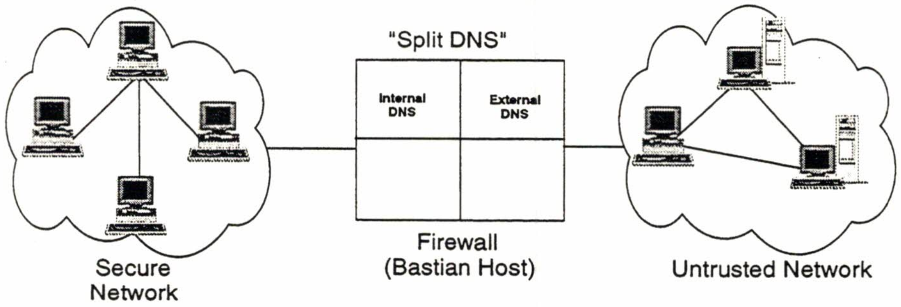
- **Socks - Defacto standard; Internet draft**
- **Uses specialized version of the application "Socksified"**
- **Client uses socks API to TCP/IP**
- **Reduces firewall load for resource intensive applications (Such as Mosaic)**

Tested Applications
- rtelnet
- rftp
- rfinger
- rwhois

- others available on Internet, but not formally tested

Copyright 1995 IBM Corporation

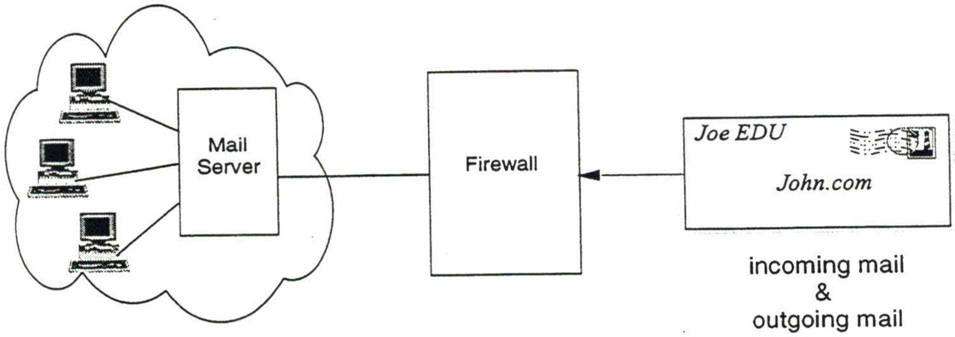
Domain Name Service (DNS)


"Split DNS"
Internal DNS
External DNS
Secure Network
Firewall
(Bastian Host)
Untrusted Network

- **External DNS reveals only Firewall host address**
- **Internal DNS reveals company host addresses to secure network**
- **Need only one valid class C address**

Copyright 1995 IBM Corporation

Mail Handling



incoming mail
&
outgoing mail

Copyright 1995 IBM Corporation

Audit

- **Authentication failures/Successes**
- **TCP/IP applications used**
  - Date/Time
  - Application
  - Incoming/Outgoing
  - Host Name
  - Userid
- **Audit records written to Syslog**

Copyright 1995 IBM Corporation

Support

- **It is provided by the experts!**
  - One-day class taught by our developers
  - Service by calling 1-800-423-5511 and speaking to our developers
  - The NetSP Forum on TalkLink on IBM Link
  - AIX Support Line

System Requirements

- **AIX/6000 3.2.5**
- **6 Mb disk space for programs**
- **50 Mb disk space for log files**
- **32 Mb of memory**
- **2 communications adapters**

CS 0704

SockCap: Client/Server Programs in Object REXX

Rick McGuire

IBM

Coding NetCentric Applications in Object REXX

Rick McGuire
Object REXX Projects
rick_mcguire@vnet.ibm.com

Why NetCentric Scripting?

- Most REXX programmers follow the rule:
- **"Script when you can -- code when you must."**
- Up until now, scripting just didn't make sense for netcentric applications and solutions:
  - Clients and Servers
  - Agent Programs
  - Web-enabled Programs
- That was then; this is now.
- **Object REXX enables NetCentric Scripting.**

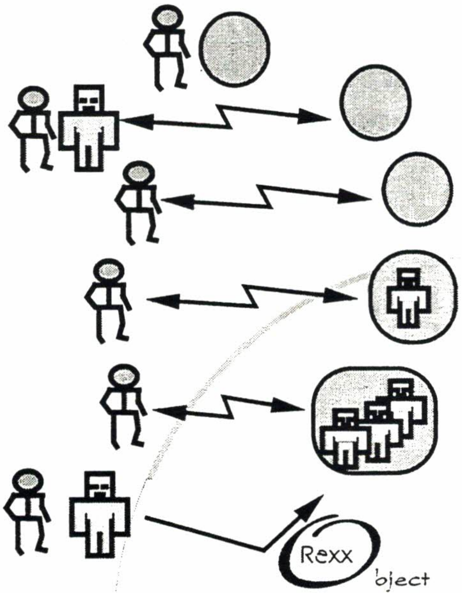
NetCentric Apps = Composite Apps

- Composite apps are built of existing programs
- Challenge: integrate a scripted solution to:
  - 1) Dial an on-line stock quote service
  - 2) Download data on two stocks
  - 3) Extract data on closing prices
  - 4) Append to stored historical data
  - 5) Create compound document summarizing
  - 6) Print, e-mail or fax to user

"Cross-Application Macro Languages", PC Magazine 2/94

(Note: Editor's Choice for OS/2 -- REXX + VX/REXX)

Modes of NetCentric Computing



- Local Application/Server
  - My word processor
- Local Agent
  - My mail filtering program
- Remote Server
  - My database server
- Remote Server with Agents
  - My Stock Brokerage Auto-Alert
- Remote Interactive Agents
  - Brokers, buyers and sellers
- Wandering Agents
  - Information Scavengers

What's New in the Language?

- Object-Oriented Programming Function
- Object-Based Concurrency
- REXX Usability Improvements
- REXX Performance Improvements

Object REXX in Six Lines

```
::Requires Dog.frm               /* frameworks & reuse */

::Class WatchDog Subclass Dog          /* inheritance */
::Method Speak                           /* extension */
  Say "WOOF"

Fang = .Watchdog~new                 /* instantiation */
Fang~Speak                               /* messaging */

==> WOOF
```

An OO View of Sockets

- "Socket": A stream of data connected from/to somewhere else
- Methods:
  - Connect: Open a stream to another socket
  - SendData: Send data over connection
  - ReceiveData: Get data from connection
  - Close: Close connection and release resources
-
  - HostName: Who's at the other end of the connection?
  - DotAddress: What's the other guy's address?
-
  - Bind: hang out my shingle on a 'known port'
  - Listen: Wait for requests on my port
  - Accept: Agree to service a client request

An OO View of Servers

- "Server": a program for hooking up sockets on demand
- How to be a Server:
  - Wait at a "known port"
  - Whenever a request for service is received,
    - Get the client a separate socket
    - Do what he wants
    - Close the client's socket
- Stateless, sessionless, one user at a time
- Examples: Echo, HTTPD
- Demos:
  - **Bounce_S.CMD (default)**
  - **Mirror_S.CMD (inherited)**

Servers with client sessions

- How to:
  - Wait at a "known port"
  - When a request for service is received,
  - Get the client a separate socket
  - **Start a ClientSession Object to service request**
    - *(Give client socket to ClientSession Object)*
- Sessions now supported in ClientSession objects
  - concurrent objects = multiple concurrent clients
  - to make a new server, make a new ClientSession class
- Examples: FTP, Telnet
- Demo: **Yodel_S.CMD**

Rexx
object

Servers with interacting clients

- How to:
  - **Start a Manager Object**
  - Wait at a "known port"
  - When a request for service is received,
  - Get the client a separate socket
  - Start a ClientSession Object to service request
  - **Hand ClientSession Object to Manager Object**
- Sessions now supported in ClientSession objects
  - concurrent objects = multiple concurrent clients
  - to make a new server, make a new ClientSession class
- Examples: MUDs
- Demo: **CHAT_S.CMD , MEETINGS.CMD**

Rexx
object

NewSocks: What We've Seen So Far...

- 4 multiuser TCP servers, with/without sessions and interaction, in 400 lines:

► *Sockets.frm*	*200 lines**
► *Servers.frm*	*50*
► *Bounce_S.cmd*	*6*
► *Mirror_S.cmd*	*12*
► *Yodel_S.cmd*	*50*
► *Chat_S.cmd*	*58*

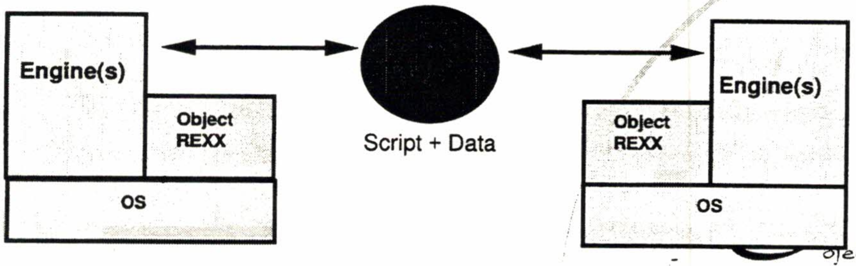
(* "Lines" = KLOC + Blank lines + Commentary)

Client/Server, without Clients?

- The World-Wide Web is based on two standards:
  - HTTP -- HyperText Transfer Protocol
  - HTML -- HyperText Markup Language
- By HTTP-enabling your servers, and serving the data in HTML, you can build C/S solutions WITHOUT specific client programs.
- So, 'web browsers' like OS/2 Web Explorer promise to become 'universal clients.' Sort of like telnet, but prettier and with more function, like pull-downs and radio buttons.
- And of course, the whole thing is easier in Object REXX...
-
- Demo: **Ticker.CMD, Auction.CMD**

"Mobile Agent" Programs

- If your program must run on another system...
  - Moving "engines" around is impractical.
  - Scripts move between systems more easily.
    - *Physically smaller than "engine" programs*
    - *Interpreted languages not HW- or OS-specific*
    - *Move state data + script = an Object*


Engine(s)
Object REXX
OS
Script + Data
Object REXX
Engine(s)
OS

Why Object REXX for NetCentric Apps

- Rich function for scripting other programs
- Modelling with concurrent objects
- Across the important platforms
- Free on most, cheap on others
- Access to SOM, OpenDoc and OS/2 Classes and Objects

Our Vision for Object REXX

- Object REXX scripting is the easiest, most productive way to integrate solutions that
  - run across multiple devices and platforms
  - move between multiple systems
  - integrate multiple applications
  - access 'heritage' programs and data
  - exploit new OO technologies and tools
  - act independently on behalf of users
- Object REXX enables NetCentric Scripting

this page intentionally left blank